MANIFOLD LEARNING FROM TIME SERIES

BY

RUEI-SUNG LIN

B.S., National Taiwan University, 1994

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Department of Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2006

Urbana, Illinois

To my parents.

ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor Prof. Stephen Levinson for his guidance and support. I sincerely thank Prof. Dan Roth who kindly serves as my de jure advisor. Without their help, I could never have the chance to complete my degree.

Many thanks to Prof. Narendra Ahuja, Prof. David Forsyth and Prof. Sylvian Ray. I am fortunate to have them in my committee, and I sincerely appreciate their suggestions and comments that help me to complete this work.

I thank to all my friends. I am grateful to have Ming-Hsuan Yang as my mentor in research. I thank Che-Bin Liu for his collaboration in learning global coordinated locally linear model. I also thank my group members, Kevin Squire and Matthew McClain, for kindly helping me with the robot experiments. In addition, I would like to thank Yen-Ju Lin, Po-Hao Chang, Alexander Kosorukov, Yu-Ping Tseng, Ying Shen and Atulya Velivelli. Their friendship provides invaluable support to me.

Most of all, I thank my parents and my sister. It is your love and encouragement that led my through these years of graduate studies.

TABLE OF CONTENTS

PAGE

LIST OF TABLES

TABLE **PAGE**

LIST OF FIGURES

FIGURE **PAGE**

CHAPTER 1

Introduction

1.1 Motivation

This thesis addresses the problem of manifold learning from time series. Usually the high dimensional data that represent a concept do not span the whole space but are embedded on a low dimensional manifold. The goal of manifold learning is to recover this low dimensional structure. While most of the manifold learning researches concentrate on developing algorithm for independently and identically distributed (i.i.d.) data. We approach the problem from a different perspective–we want to learn the manifold from temporal sequences. Our motivation is based on the observation that in the many real world applications the data available are actually time series, and they are strong temporal dependency among these data samples. Under such circumstances temporal information should be taken into consideration.

Learning manifold from sequences is a challenging task. As will be pointed out in the following chapter, almost all manifold learning algorithms for i.i.d. samples are unable scale up to include temporal information. In addition, including temporal dynamics implies a more complicated model has to be used which makes the learning difficult.

In our work, we parameterize the manifold using locally linear models and extent model along the time to account for the temporal correlation between observations. The result is a dynamic Bayesian network in the form nonlinear state-space model. Exact inference and learning on this graphical model is intractable. However, we show that based on the locally linear property of our model, we are capable of providing efficient approximate inference and learning algorithms. The experimental results confirm the correctness of our model, and it also demonstrates that by taking into account temporal information help us achieve better learning results.

Under our proposed model, we provide a unified framework that can perform manifold learning from time series and dynamic inference on a manifold . Even more surprisingly, our model share great resemblance to the Kalman filter. Yet our model is apparently more general, since our model preform inferences on a nonlinear manifold.

We then apply our manifold learning algorithm to real world learning problems, including robot localization and visual tracking. We are particularly interested in the map learning problem. Since we are the automatic language acquisition research group, our long term goal is to develop an autonomous mobile robot capable of learning spoken language through acting in the real world. Letting the robot acquiring the concept of space will certainly be a milestone in our research.

Because our purpose of using the map is different from most robotic researches, we formulate the mapping learning problem differently. We do not rely on the sensors on the robot to provide any distance measurement between the robot and the observed object.

On the other hand, we plan to acquire the nonlinear mapping between the sensory input and the robot's location. This is clearly manifold learning problem.

We formalize the map learning problem as follows. The robot is let to wonder in the environment while its observation and action sequences are recorded. We then apply our proposed learning algorithm to consolidate the robot's navigation experiences into a nonlinear manifold. This manifold then represents the conceptual map of the environment. This is apparently a difficult task. Especially, given the fact our robot's odometer readings contain large noises. Yet our experimental results demonstrate that our map learning algorithm correctly acquires a topological map of the environment.

1.2 Thesis Outline

This thesis is organized as follows.

We start with a survey of manifold learning in Chapter 2. We provide an overview of the manifold learning problem and then emphasize on the methods that will frequently be inferred in the following chapters. In this chapter, we also examine the extensibility of each method for learning temporal sequences, and justify our choice of using locally linear model for manifold learning.

In Chapter 3, we first review a locally linear model, the probabilistic principal component analyzer, and its extension, the mixture of probabilistic principal component analyzers. Our model is based on mixture of probabilistic principal component analyzers, so we provide the detailed description of inference and learning in this model. On the second part of the Chapter, we focus on a single probabilistic principal component

analyzer and present our online learning and online discriminative analysis algorithms for it. We address these two algorithms since we believe they will become be the foundation of learning manifold online.

In Chapter 4 we introduce the global coordination model, which our manifold model is based on. The global coordination model is a mixture of probabilistic principal component analyzers with additional alignment on the latent variables of PCA coefficients. The alignment is critical for mapping high dimensional observations to low-dimensional global coordinated vectors. Learning the alignment is a ill-posed problem. We introduce the three representative algorithms for learning the global coordination model from i.i.d. samples and their limitations. It is because there algorithms fail to provide satisfactory results that motivate us to pursue learning global coordination model from time series.

Chapter 5 is the focus of this work, in which we introduce our manifold learning algorithm from time series. We formulate the learning problem as learning a state-space model. The exact inference on this model is intractable, so we provide an approximate inference algorithm for it. Based on which, we show that the model can be learned through a variational method. We then experiment on our model using synthetic data. The result shows that our model achieves superior results than those algorithms learn from i.i.d samples.

In Chapter 6 we apply our model to two real world applications: object tracking and robot map learning, and we conclude our work in Chapter 7.

1.3 Contributions

The contribution of this thesis from conceptual level to the model detail is highlight here.

- We address the plausibility of learning manifold from time series, while most the existing manifold learning algorithms focus on learning manifold from i.i.d. sample.
- We present a novel algorithm for manifold learning from time series and our experiment on synthetic data confirms that including the additional temporal information information can improve the learning result.
- Our model is an extension of state-space model and it provides a unified framework for manifold learning from sequences and dynamic inference on the manifold. That is, the same model we use to learn the manifold can be used for dynamic inference directly.
- Our inference and learning procedures share a great similarity with Kalman filter with little computation overhead. Yet our model is more general than Kalman filter. While Kalman filter performs dynamic inference on a linear manifold, our model perform inference on a nonlinear manifold. This make our proposed model a very attractive extension of Kalman filter.
- We propose a novel robot map learning algorithm based our state-space model. Unlike most map learning algorithms which rely on fine sensors and odometers to provide fine distance measurement. Our algorithm learns the nonlinear mapping

from sensory input to the robot's location directly, and we have obtained promising results.

CHAPTER 2

Manifold Learning

In the chapter, we introduce the concept of manifold. We provide an overview on manifold learning in Section 2.1, and explain the detail algorithms of the selected models in the subsequent sections.

2.1 Introduction

Most of the data we perceive have more correlated features than their true degree of freedom. Geometrically, this can be interpreted as a curved manifold lying in a high-dimensional space. Take the images of a human face as an example. If each face image is represented as an array of pixel-based features, these images will correspond to a set of points in this high dimensional feature space. However, the variation in these face images is determined by a small number of causes, such as the viewing angles, the illumination conditions, and facial expressions, etc. In addition, there are strong correlation between these images; i.e. images taken under similar conditions will be close to one another in the feature space. These properties lead to the conclusion that the images of a human

face are distributed along a low-dimensional manifold in the high-dimensional pixel-based feature space.

Manifold learning refers to all the methods that recover the low-dimensional intrinsic structures of a manifold from a set of high dimensional observations. While all these algorithms share the same concept of manifold, their manifold representation might be different from one another depending on their applications. Here we divide the manifold learning problems into three types:

- *Type I:* measuring the Euclidean distance from a sample to the manifold.
- *Type II:* measuring the geodesic distance between samples on the manifold.
- *Type III:* interpolating between samples on the manifold.

Algorithms for the type I manifold learning problem are mainly used in the detection and classification applications. If a class is modeled as a manifold, the distance from a given sample to the manifold is an indicator of whether the sample belongs to this class. For these applications, the manifolds are usually parameterized using mixtures of linear models, such as mixtures of principal component analyzers [1][2][3][4] and mixtures of factor analyzers [5][6]. There are limitations on these mixture models. Although they suffice for the classification problems, they are unable to determine the geodesic relationship among samples lying on the manifold.

The goal of the type II manifold learning problem is to derive a mapping that converts high-dimensional observations to low-dimensional intrinsic parameters such that the geodesic distances between samples in the observation space correspond to the their

Euclidean distances in the parameter space. This is a dimensionality reduction problem. If the manifold is linear, the linear mapping function and can be efficiently computed using principal component analysis (PCA) [7], or factor analysis [8]. When the manifold is nonlinear, the linear models have to be generalized to cover the nonlinearity. There are various approaches, including the principal curves and its variants [9][10] which look for a curve or surface that passes through the middle of data set, the Gaussian process latent variable model [11] which formulate PCA as a Gaussian process with particular prior on model parameters, or the kernel PCA [12] which apply kernel functions on PCA models.

Nonlinear embedding [13][14][15][16] is another approach for the type II learning algorithms. These methods directly convert a set of high dimensional observations to a set of low dimensional intrinsic parameters without resorting to any mapping function. Since these methods do not derive mapping functions, for samples that are no included in the data set, their intrinsic parameters are unknown. This is referred as the out-of-sample problem in nonlinear embedding. As a result, nonlinear embedding methods are valid tools to analyze or visualize high dimensional data, but they have limited applications. Recently, an solution for the out-of-sample problem has been proposed by Bengio et. al. [17]. They show that nonlinear embedding methods can be regarded as the variants of the Kernel PCA algorithms with particular kernel functions. Therefore, the nonlinear mapping functions for these embedding algorithms can be derived through kernel PCA. Kernel PCA is a powerful techniques for nonlinear dimensionality reduction, but it only provides a unidirectional mapping (from high dimensional observation space to low-

dimensional parameter space), which make it incapable of computing the interpolation on the manifold.

Type III manifold learning algorithms require a bidirectional mapping function, based on which the interpolation on manifold works as follows. Firstly, the points for interpolation are mapped from the observation space to the intrinsic parameter space. Then, the linear interpolation is the performs using these mapped intrinsic parameters. Finally, the interpolation result in the parameter space are lifted back to the observation space. The requirement of a bidirectional mapping function makes type III the most difficult manifold learning problem among the three.

The bidirectional mapping can represented using either a global nonlinear model or a piecewise linear model. Elgammal et. al. [18] use an invertible Generalized Radial Basis Function to model this bidirectional nonlinear mapping. Neil [11] use Gaussian process to estimate the joint probability of the observation and the intrinsic parameters, from which the bidirectional mapping can be derived. The piecewise linear models for bidirectional mapping are based on mixture of factor analyzers [8] analyzers [5]. These mixtures models alone can not provide mappings because the latent variables in each mixture component use their own local coordinates. Roweis et. al. [19] show that by aligning the latent variables in the mixture components in a globally coordinated system, this piecewise linear model can provide a bidirectional mapping. Based on the same global coordination idea, Brand[20], Teh et. al. [21] and Verbeek et. al. [22] provides alternative manifold algorithms.

All the algorithms above assume that samples in the data set are independently, identically, distributed (i.i.d). If samples in the data set are not i.i.d. but sequences where there are strong temporal correlation, this temporal information can provide additional cues for manifold learning. One of the earliest work on manifold learning from sequences is by Bishop [23], who provides an algorithm to learn the generative topological mapping (GTM) from time series. The generative topological mapping (GTM) [24] is a probabilistic version of self-organizing map [25] which can be regarded as a manifold learning algorithm. Brand [26] parameterizes the manifold using a Hidden Markov Model [27] in which a each state covers a local region of the manifold and the state transition table describes the geodesic relationships between this states. Therefore, learning the manifold is equivalent to learning the HMM from sequences. Rahimi et. al. [28] proposes a semi supervised algorithm to learn the unidirectional nonlinear mapping from the observations to the intrinsic parameters. They formulate this manifold learning problem as optimization problem by assuming the observed sequences are generated from a linear dynamic model. Learning the nonlinear embedding from sequences is first proposed by Jenkin et. al. [29]. This idea of spatial-temporal embedding is then adopted by Bowling et. al.[30] who learn a nonlinear embedding using the action-observation sequences obtained from a robot. To learn a bidirectional mapping function from sequences, Wang et. al. [31] extend Neil's Gaussian process latent variable model[11] to become a Gaussian process dynamic model in which the dynamics can be nonlinear.

In this work, our focus is on manifold learning using mixtures of piecewise linear models. The mixture of piecewise linear models is an attractive manifold learning algo-

rithm since with global coordination on its latent variables it can solve all three types of manifold learning problems. We propose a novel manifold learning algorithm that learns from data sequences. Our model is based on the global coordination model proposed by Roweis [19]. The global coordination model is designed to learn from i.i.d. samples, which is ill-posed and has serious local minimal problem. We expand the global coordination model to become a dynamic Bayesian network and provide inference and learning algorithm of this model. Experimental results demonstrate that by taking temporal information into account, our algorithm achieve better results. We then apply the model to dynamic visual problems including object tracking, video synthesis and robot map building. We demonstrate that the global coordination model provide an practical and efficient approach to perform dynamic inference on a nonlinear manifold. All these details will be introduce in the following chapters.

In the remaining of the chapter, section 2.2 reviews nonlinear dimensionality reduction using kernel PCA. Section 2.3 first reviews the two nonlinear embedding methods, the ISOMAP [13] and the LLE [14]. It then demonstrates that kernel PCA and nonlinear embedding methods merges under the Nystrom formula [17]. Section 2.4 provides a brief introduction on the global coordination of piecewise linear model, and address the problem incorporating temporal information to improve learning. Following this is the summary of the chapter.

2.2 Nonlinear Dimensionality Reduction

In this section, we review the dimensionality reduction methods based on principal component analysis (PCA). We start with linear dimensionality reduction using PCA. Then, we demonstrate that PCA can be generalized for nonlinear dimensionality reduction by the using kernel functions. As will be shown in the following sections, kernel PCA is closely related to nonlinear embedding methods.

2.2.1 Principal Component Analysis

Principal component analysis (PCA) is a widely used technique to extract structures from high dimensional data set. It linearly projects the data set to a low dimensional space while preserving the maximal variances of the data. As a result, PCA is a linear projection with minimal reconstruction error.

Let $X = \{x_1, x_2, \ldots, x_N\}$ be a data set of d-dimensional vectors. The covariance matrix of X is defined as:

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T \tag{2.1}$$

where $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the sample mean. Let u be an eigenvector of S, $Su = \lambda u$. PCA chooses the q most significant eigenvectors of S as the linear projection matrix $W = (u_1, \ldots, u_q)$. The eigenvectors of S can be computed by applying singular value decomposition [32] to matrix, $M = (x_1 - \bar{x}, x_2 - \bar{x}, \ldots, x_N - \bar{x})$.

Let z_i be the linear projection of sample x_i, z_i is defined as:

$$z_i = W^T(x_i - \bar{x}) \tag{2.2}$$

Given z_i, the reconstruct of x_i is:

$$\tilde{x}_i = W z_i + \bar{x} \tag{2.3}$$

It can be proved that since PCA retains the axes in the sample space where S has the largest variances, PCA minimizes the reconstruction error $\sum_{i=N}^{T} ||x_i - \tilde{x}_i||^2$.

PCA provides an optimal solution to model linear manifold, and it provides bi-directional mapping between high dimensional sample space and low-dimensional vector space. However, for a nonlinear manifold, PCA fails to extract the underlying intrinsic structures.

2.2.2 Kernel Principal Component Analysis

To apply PCA for nonlinear manifold learning, we have to generalize the PCA model. Here we introduce kernel PCA [12], which generalizes PCA model using the kernel function. Kernel PCA uses a nonlinear function to map data samples to an arbitrary high dimensional feature space, in which the samples are expected to be embedded in a linear subspace. It then apply PCA in this high dimensional space.

Let $\Phi : \mathcal{R}^d \rightarrow \mathcal{R}^D$ be the nonlinear function that maps data samples from the d-dimensional sample space to a D-dimensional feature space. To simplify the analysis,

we assume that the sample mean in the feature space is zero, $\frac{1}{N}\sum_{i=1}^{N}\Phi(x_i) = 0$. The covariance matrix in the feature space becomes:

$$C = \frac{1}{N}\sum_{i=1}^{N}\Phi(x_i)\Phi(x_i)^T \tag{2.4}$$

The task now is to find the q most significant eigenvectors of C.

Let V be the most significant eigenvector of C. V is known to lie on the span of $\{\Phi(x_1), \Phi(x_2), \ldots, \Phi(x_N)\}$, so it can be described as:

$$V = \frac{1}{N}\sum_{i=1}^{N}\alpha_i\Phi(x_i) \tag{2.5}$$

Since V is a eigenvector of C, we can obtain the equation below:

$$\phi(x_k)^T CV = \lambda\phi(x_k)^T V \tag{2.6}$$

where λ is the largest eigenvalue of C. Combining equation (2.1), (2.5) and (2.6), we will get:

$$\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(\alpha_j\Phi(x_j)\cdot\Phi(x_i)\right)\left(\Phi(x_k)\cdot\Phi(x_i)\right) = \lambda\sum_{j=1}^{N}\alpha_j\left(\Phi(x_k)\cdot\Phi(x_j)\right) \tag{2.7}$$

Denote K as a $N \times N$ matrix in which $K_{ij} = (\Phi(x_k)\cdot\Phi(x_j))$ and define α as a vector with $\alpha = (\alpha_1, \ldots, \alpha_N)^T$. Equation (2.7) can be written as:

$$\frac{1}{N}K^2\alpha = \lambda K\alpha \tag{2.8}$$

Hence, α is the most significant eigenvector of K, $K\alpha = N\lambda\alpha$, and $N\lambda$ is the corresponding eigenvalue. To ensure that $V^T V = \alpha^T K \alpha = 1$, we have to re-scale α as:

$$\alpha = \sqrt{\frac{1}{N\lambda}}\alpha \tag{2.9}$$

In Kernel PCA, Φ is never actually computed. All the computations are based on the dot product:

$$k(x, y) = (\Phi(x) \cdot \Phi(y)) \tag{2.10}$$

where k is a kernel function. An example of the kernel is the radial basis functions [33]:

$$k(x, y) = \exp\left(\frac{||x - y||^2}{2\sigma^2}\right) \tag{2.11}$$

Let $\{\alpha^1, \ldots \alpha^q\}$ be the q most significant eigenvectors of K that have been rescaled according to (2.9). Let $v = (v_1, \ldots, v_q)^T$ be the nonlinear projection of x using kernel PCA:

$$v_j = \sum_{i=1}^{N} \alpha_i^j \left(\Phi(x_i) \cdot \Phi(x)\right) = \sum_{i=1}^{N} \alpha_i k(x_i, x) \tag{2.12}$$

It is worth to note that kernel PCA no longer provides bi-directional mapping between x and v, since it can not map data from low-dimensional vector space to the sample space. This is because the inverse function of Φ is unknown. There is another limitation of kernel PCA. From equation (2.12), computing the projection of a new sample requires all the

training samples, which means all the training samples have to be saved in the memory and the computational complexity grows linearly with the number of training examples. This makes kernel PCA less favorable for time critical applications.

Kernel functions define the dot product in the feature space. The choice of kernel function determines the results of kernel PCA. In the following sections, we will introduce a different approach for manifold learning termed nonlinear embedding. Then we will show that nonlinear embedding methods are actually kernel PCAs with particular kernel functions.

2.3 Nonlinear Embedding

Nonlinear embedding takes in a data set and map all the samples in the data set to a low dimensional global-coordinated space. Embedding methods are not actual mapping functions, since they do not provide mapping for samples not included in the data set. Here we review two seminal works on nonlinear embedding, the ISOMAP [13] and LLE [14]. We then demonstrate their connections with kernel PCA.

2.3.1 ISOMAP

Given a set of high dimensional samples $X = \{x_1, \ldots, x_N\}$, ISOMAP computes a low-dimensional embedding by minimizing the distortion of geodesic distance between samples. The actual geodesic distances among samples are unknown, but when the samples are dense, the geodesic distances between a given sample x_i and its neighbors

The ISOMAP algorithm

Given data set $X = \{x_1, \ldots, x_N\}$,

1. *Construct neighborhood graph*: Create a graph of N vertices that correspond to the N samples. For each samples x_i, find its k nearest neighbors. Add edges to the graph to connect the sample and it neighbors, and set the weights to be the Euclidean distances between them.

2. *Compute pairwise shortest path*: Based on the neighborhood graph, apply the Floyd-Warshall algorithm to compute all-pair shortest paths and save the result in D_x.

3. *Compute the embedding*: Find Y that minimizes (2.13) by performing eigen decomposition on similarity matrix S.

Table 2.1 Details of the ISOMAP algorithm algorithm

can be estimated using the Euclidean distances between them. Based on which, we can then estimate the pairwise geodesic distances between samples.

The detail works as follows: given a data set of N samples, a graph of N vertices is constructed where each sample represents a vertex. For each sample, its k nearest neighbors are located and edges are added to connect this sample with its neighbors. Each edge has a weight which is the Euclidean distance between the two vertices. Now given this graph, we can compute the all-pairs shortest paths using the Floyd-Warshall algorithm [34]. The pairwise shortest distances among vertices will be the estimates of their geodesic distances.

Denote D_x as a $N \times N$ matrix in which entry $D_x(i, j)$ contains the estimated geodesic distance between sample x_i and sample x_j. Let $Y = [y_1| \ldots |y_N]$ be a matrix in which y_i is the q-dimensional nonlinear embedding of x_i, and let D_y be another $N \times N$ matrix with $D_y(i, j) = ||y_i - y_j||$. For all possible i and j, we want the Euclidean distances

between y_i and y_j to be close to the geodesic distance between x_i and x_j. By applying classic multidimensional scaling (MDS), this constraint can be described as :

$$E = ||\tau(D_x) - \tau(D_y)||_{L_2} \tag{2.13}$$

where τ is a function that convert Euclidean distances to inner products, $\tau(D) = -HMH/2$, with $M_{ij} = D_{ij}^2$ and $H_{ij} = \delta_{ij} - 1/N$. The ISOMAP algorithm finds Y that minimizes E in (2.13).

Denote similarity matrix $S = \tau(D_x)$. It can be shown that Y is determined by the q most significant eigenvectors and eigenvalues of S. Denote e_i and λ_i be the i-th most significant eigenvector and eigenvalue of S respectively. Then, $Y = (\sqrt{\lambda_1}e_1, \sqrt{\lambda_2}e_2, \ldots, \sqrt{\lambda_q}e_q)^T$. We summarize the ISOMAP algorithm in Table 2.1.

2.3.2 Locally Linear Embedding

Locally Linear Embedding (LLE) is another seminal nonlinear embedding method . Unlike ISOMAP, LLE attempts to preserve local geometrical structure between samples after the embedding. It is based on the assumption that with sufficient data $X = \{x_1, \ldots, x_N\}$, any given sample x_i and its k neighbors will lie on a linear patch. Therefore, sample x_i can be linearly reconstructed from its neighbors,

$$x_i \approx \sum_{j \in L_i} w_{ij} x_j, \quad \forall i = 1, \ldots, N. \tag{2.14}$$

where set L_i contains indices of x_i's k-nearest neighbors.

Locally Linear Embedding

Given data set $X = \{x_1, \ldots, x_N\}$,

1. *Select neighbors*: for each samples x_i, find its k nearest neighbors and store the result in L_i.
2. *Compute Reconstruction weights*: Compute W according to (2.15).
3. *Compute the embedding*: Compute Y according to (2.16).

Table 2.2 Details of Locally Linear Embedding algorithm

LLE first construct a $N \times N$ matrix W from data set X that minimizes the equation:

$$\epsilon(W) = \sum_i |x_i - \sum_{j \in L_i} W_{ij} x_j|^2 \tag{2.15}$$

in which W_{ij} is set to zero if x_j is not among the k-nearest neighbors of x_i and there is a constraints on each row of W, $\sum_j W_{ij} = 1$. This is a constrained least square problem with close-form solution [35]. After W is known, we can then compute Y, the nonlinear embedding of X. LLE requires any sample y_i can be approximately reconstructed from its neighbors using the weights defined in W. Denote matrix $Y = [y_1|\ldots|y)N]$, Y has to minimize the following equation:

$$\Phi(Y) = \sum_i |y_i - \sum_j W_{ij} y_j|^2 = Y(I - W)^T(1 - W)Y^T \tag{2.16}$$

By assuming $\sum_i y_i = 0$ and $\sum_i y_i {y_i}^T = I$, it can be shown that Y, the q-dimensional embedding of X, is determined by the least significant eigenvectors of S, with $S = (I - W)^T(1-W)$. Let f_i be the i-th least significant eigenvector of S, $Y = (f_j^2, f_j^3, \ldots, f_j^{q+1})^T$. The complete LLE algorithm is summarized in Table 2.2

Note that the solutions in kernel-PCA, ISOMAP and LLE all involve the performing eigen decomposition on a $N \times N$ matrix. In the next section, we show that these three methods can be unified in a framework.

2.3.3 Out-of-Sample Embedding

ISOMAP and LLE are not actual mapping functions. Given the nonlinear embedding result of data set X, neither ISOMAP nor LLE can infer the embedding of samples that are not included in X. This out-of-sample problem has greatly limited the applications of both methods. A straightforward solution is to expanding the data set X by adding in the new samples and compute the embedding on the new data set [36][37]. Since the nonlinear embedding is recomputed every time new samples are available, a lot of redundant computation is involved. This approach is apparently not practical for time critical real world applications.

An alternative approach is to learn a nonlinear mapping function of ISOMAP or LLE from the original data set X [17]. In the ISOMAP algorithm, we have a $N \times N$ similarity matrix $S = \tau(D_x)$. If we can find a kernel function $s(\cdot)$ such that $S_{ij} = s(x_i, x_j)$, the ISOMAP algorithm becomes a kernel PCA with the particular function $s(\cdot)$. Let α_i and λ_i be the i-th most significant eigenvectors and eigenvalues of S respectively and $\alpha_{i,j}$ be the j-th entry in α_i. The q-dimensional nonlinear embedding of X using ISOMAP is:

$$\left\{ y_i = \left(\frac{1}{\sqrt{\lambda_1}} \sum_{j=1}^{N} \alpha_{1,j} s(x_j, x_i), \frac{1}{\sqrt{\lambda_2}} \sum_{j=1}^{N} \alpha_{2,j} s(x_j, x_i), \ldots, \frac{1}{\sqrt{\lambda_q}} \sum_{j=1}^{N} \alpha_{q,j} s(x_j, x_i) \right)^T \right\}_{i=1}^{N} \tag{2.17}$$

Let $Y = [y_1 | \ldots | y_N]$. We can rewrite Equation (2.17) as:

$$Y = \left(\frac{1}{\sqrt{\lambda_1}} S\alpha_1, \frac{1}{\sqrt{\lambda_2}} S\alpha_2, \ldots, \frac{1}{\sqrt{\lambda_q}} S\alpha_q \right)^T \tag{2.18}$$

Since $S\alpha_1 = \lambda_i \alpha_1$, equation (2.18) is consistant with the ISOMAP algorithm described in Section 2.3.1. For data sample x that is not included in the data set X its nonlinear embedding y can now be computed as:

$$y = \left(\frac{1}{\sqrt{\lambda_1}} \sum_{j=1}^{N} \alpha_{1,j} s(x_j, x), \frac{1}{\sqrt{\lambda_2}} \sum_{j=1}^{N} \alpha_{2,j} s(x_j, x), \ldots, \frac{1}{\sqrt{\lambda_q}} \sum_{j=1}^{N} \alpha_{q,j} s(x_j, x) \right)^T . \tag{2.19}$$

Therefore, by reformulating the ISOMAP algorithm as a kernel PCA problem, we obtain the mapping function for ISOMAP. The same trick can be applied to LLE. Note that the above method lies on the premise that such kernel functions $s(\cdot)$ exist for ISOMAP and LLE. Bengio et. al. [17] propose two approximate kernel functions for ISOMAP and LLE accordingly.

This section demonstrates that nonlinear embedding methods such as ISOMAP and LLE can become a nonlinear dimensionality reduction functions, if they are formulated as kernel PCA problems. Defining the kernel functions for nonlinear embedding methods

is a challenging problem as is shown in [17]. In addition, like kernel PCA, computing these nonlinear mappings are computational and memory intensive.

2.3.4 Spatio-Temporal Embedding

ISOMAP and LLE perform nonlinear embedding on data sets in which samples are independently, identically,distributed (i.i.d.). Assuming data set $X = (x_1, x_2, \ldots, x_N)$ is now a sequence whose trajectory forms a smooth curve on the manifold. Let $Y = (y_1, y_2, \ldots, y_N)$ be the embedding of X, we expect that there are strong local temporal correlations among the samples in Y; that is sample y_i should be close to its temporal neighbors. In other words, in this new data set besides the local spatial relationships among samples based on which the ISOMAP and the LLE algorithms are developed, there are additional temporal relationships among samples. Taking into account this temporal information could certainly improving the embedding results especially when data samples are sparse.

The spatio-temporal by Jenkins et. al. [29] is the pioneer work to address problem. They follow the ISOMAP algorithm describe in Table 2.1 to construct a neighborhood graph according to the local spatial properties among the samples. They then modify this neighborhood graph by scaling down the weights of the edges that connect a sample to its temporal neighbors so as to ensure the geodesic distance between temporal neighbors are small. The similarity matrix which contains the pairwise geodesic distances between samples are then estimated based on this modified neighborhood graph, and the following steps are the same as those in ISOMAP. In this spatio-temporal algorithm the scaling

factor used for reweighting the neighborhood graph is manually chosen. In addition, the temporal properties among samples imposed in the algorithm rudimentary. These unsatisfactory results can be attributed to the fact that it is difficult combine spatial and temporal information together in the similarity matrix. There is still no feasible solution that does not resort to heuristics.

Weinberger et. al. [38] present an algorithm that formulates the nonlinear embedding result as the solution of a constrained optimization problem. The constraints of the optimization problem are derived according local spatial geometry among samples, and the optimization problem is solved using semi-definite programming. Following the same framework, Bowling et. al. [30] propose an action-respecting algorithm that computes nonlinear embedding from a data sequence. Their application is on a robot map learning experiment, and they derive temporal constraints based on the robot's actions. Their optimization problem contains both spatial and temporal constraints, and the solution can still be obtained through semidefinite programming.

2.4 Piecewise Linear Models

The idea of using piecewise linear model to approximate an nonlinear function has been well perceived. Schaal et. al. [39] use a mixture of locally linear models to solve a nonlinear regression problem. Manifold learning is more difficult than nonlinear regression. Unlike a regression problem where each training sample is an input-output pair of the nonlinear function, the training data set in manifold learning contains only the

outputs. Therefore, a manifold algorithm has to estimate the corresponding inputs and compute the nonlinear function at the same time.

Piecewise linear models, such as mixtures of principal component analyzers, parameterize a nonlinear manifold by dividing the manifold into locally linear regions. The parameterization, however, does not provide an nonlinear mapping to a global coordinate parameter space since each locally linear model has its own coordinate system. Roweis et. al. [19] shows that by linearly aligned these mixture components in a global coordinate system, this extended model can provide bidirectional mapping between observations and global coordinate intrinsic parameters. They call this the global coordination (GC) model.

Learning a GC model is a ill-posed problem, because the low dimensional global coordinate intrinsic parameters are unobservable. Therefore, additional constraints on the alignment have to be imposed. These constraints can be derived from either the nonlinear embedding results or the geometric properties of the mixture model. Wang et. al. [6] use the ISOMAP result as the intrinsic parameters, and formulate the learning problem as a regression problem. Teh et. al. [21] formulate it as an convex optimization problem such that the alignment result should be minimally distorted from the LLE result. On the other hand, Roweis et. al. [19] and Verbeek et. al. [22] impose the spatial constraints on adjacent mixture components. All the algorithm above use i.i.d data set.

There are limitations on each of these algorithms. [6] and [21] need good nonlinear embedding result from either the ISOMAP or the LLE. Since both the ISOMAP and the LLE algorithms use spatial nearest neighbors to compute the embedding, their results

are sensitive to the data set used. If the data are sparsely sampled, there would be a lot of noise in the spatial neighbors which will deteriorate the embedding results. It is also difficult derive spatial constraints on alignment directly from the mixture models. The spatial constraints use in [19] an [22] can only be satisfied when a large number of mixture components are used.

2.4.1 Learning Global Coordination Model from Time Series

Learning the GC model from i.i.d. samples is hard. However, if the data set are sequences of samples, the additional temporal information could improve the learning result. As a matter of fact, in many real applications observations are perceived in sequences, which makes learning the GC model from sequences a practical problem.

Learning the GC model from data sequences has never been explored before. In this work we propose a novel algorithm to learning the GM model from sequences of data. We expand the GC model to become a dynamic Bayesian network, and like the earlier work of manifold learning form sequences by Bishop [23] and Brand [26] we use the forward-backward like algorithm to estimate hidden variables and compute model parameters. Unlike [23] and [26] use discrete state variables, our state variables are continuous vectors. Because the exactly inference on our dynamic GC model is intractable, we propose an approximate inference algorithm using variational methods[40]. Based on the this inference algorithm, we then derive the learning algorithm of our dynamic GC model. The detail of our algorithm is explained in Chapter 5 and the experiments of our model is presented in Chapter 6.

2.5 Summary

In this chapter, we review the manifold learning problem. We divide the manifold learning algorithms into three types with each type focus on particular applications. We also explain selected manifold learning algorithms in details. The literature review in the Chapter is not to be comprehensive, but to lay the foundation for the following discussion.

Chapter 3 reviews the local linear model based on probabilistic principal component analyzer (PPCA). It also present our adaptive learning algorithm for PPCA. Chapter 4 introduces the global coordination model which is an extension of the mixture of local linear models. Chapter 5 presents our proposed algorithm to learn the GC model from data sequences. Following that are the chapters for the experiments and the conclusion.

CHAPTER 3

Local Linear Models

In this chapter, we present a particular local linear model termed probabilistic principal component analysis (PPCA) [41], and the mixture model based on it. The two models become the foundation of our global coordination model which will be introduced in the following chapters.

PPCA formulates principal component analysis [7] as a latent variable model with Gaussian noise. By imposing Gaussian prior on the latent variables, PPCA becomes a probabilistic model for data distributed on a local region in a linear subspace. It is because the local modeling property that PPCA model serves as a valid candidate to parameterize a nonlinear manifold. A nonlinear manifold can be divided into a number of piecewise linear regions, and each locally linear region can then be parameterized using a single PPCA model. Hence, a nonlinear manifold can be parameterized by a mixture of PPCA models [5].

Both PPCA and mixtures of PPCAs are generative models, so learning the models are through inference algorithms. Here we provide not only their learning algorithms but also the geometrical interpretation of these inference procedures. We will illustrate that because of its innate structure mixtures of PPCA models can only provide limited

inferences on nonlinear manifolds. This motivates the development of global coordination model introduced in the next chapter.

In the second half of this chapter, we focus on a single PPCA model. We first study the problem of adaptively adjusting a single PPCA model according to new data samples. We envision a problem that we are moving a local linear patch along a nonlinear manifold and collecting new data samples along the way. The goal is to adaptively adjust this local linear patch according on the incoming data stream and to make the local patch fit the manifold as good as possible. This abstract problem has many practical application, such as visual tracking, as will be seen in the following chapters.

Incrementally learning/updating principal component analysis has been well studied, and efficient algorithms based on singular value decomposition [32] have been proposed [42][43][44]. However, all these algorithms assume the mean of the data stream to be zero. This is certainly not true for the abstract problem we just described. In this chapter, we present a novel algorithm to adaptive adjust probabilistic PCA without the zero-mean assumption.

The last topic in chapter is discriminative analysis on PPCA, which applies to situations when PPCA is used for detection problems and negative examples are available. We present a novel algorithm to take into account those negative examples. Our algorithm performs discriminative analysis on the existing generative model, and can be applied to dynamic inference problems. We will illustrate the practical applications of this model in Chapter 6.

3.1 Probabilistic Principal Component Analysis

Probabilistic principal component analysis (PPCA) is a latent variable model with Gaussian noise [45][41]. Let y be a d-dimensional observation data generated by q-dimensional latent variables z with $(d > g)$. Define correspondence between y and z as:

$$y = Wz + \mu + \epsilon \tag{3.1}$$

where μ is a constant vector, W is a $d \times q$ matrix, and ϵ is a d-dimensional zero-mean Gaussian noise, $\epsilon \sim \mathcal{N}(0, \sigma^2 I_d)$. Conventionally, latent variables z are set to be independent Gaussians with zero means and unit variances:

$$P(z) \sim \mathcal{N}(0, I_q) \tag{3.2}$$

This completes the definition of the model.

Given a set of observation $Y = \{y_1, y_2, \ldots, y_N\}$, learning PPCA is to estimate model parameters, $\Theta = (\mu, W, \sigma^2)$ from Y through maximum likelihood estimation (MLE) [46]. In the following, we will first introduce the inference processes in PPCA. Based on the inference results, we then derive the learning algorithms. As will be shown in the following, the subspace spanned by PPCA is identical to PCA model.

3.1.1 Inference

Once model parameters Θ are known, a number probabilistic inferences can be performed on the PPCA model. Firstly, from equation (3.1), it is straight forward to derive $P(y|z)$ as a gaussian:

$$P(y|z) \sim \mathcal{N}(Wz + \mu, \sigma^2 I_d) \tag{3.3}$$

With $P(y|z)$ known, together with equation (3.2), we can compute

$$P(y) = \int P(y|z)P(z)dz \sim \mathcal{N}(\mu, WW^T + \sigma^2 I_d) \tag{3.4}$$

which is the likelihood of observation y generated from the model. In addition, we can also infer z from y:

$$P(z|y) = \frac{P(y|z)P(z)}{P(y)} \sim \mathcal{N}((W^T W + \sigma^2 I_q)^{-1} W^T (x - \mu), \sigma^2 (W^T W + \sigma^2 I_q)^{-1}) \tag{3.5}$$

Equation (3.3) and equation (3.5)provide the mappings between observation x and latent variables z. Denote f as the function that maps from z to y. We define f as: $f(z) = E[P(y|z)]$. Similarly, we set the function that maps from y to z as: $g(y) = E[P(z|y)]$. It is worth to note that the mapping in PPCA is not self-consistent.

$$f(z) \neq E[y|g(y) = z] \tag{3.6}$$

This is because the imposed Gaussian prior on the latent variables pulls the mapping $g(y)$ toward the origin. A detailed explanation of this phenomenon is available in section 3.1.3.

Even with this limitation, PPCA is still a very powerful model and has many practical applications. Our global coordination model is also based on PPCA.

3.1.2 Learning

With the inferences equations known, learning PPCA model is based on maximum likelihood estimation. Given data set $Y = \{y_1, y_2, \ldots, y_N\}$, the optimal model parameters Θ^* is:

$$\Theta^* = \arg\max_{\Theta} \log P(Y|\Theta) = \arg\max_{\Theta} \sum_{i=1}^{N} \log P(y_i|\Theta) \tag{3.7}$$

Since samples in Y are assumed to be independently, identically, distributed (i.i.d). Based on equation (3.4), it can be shown that the objective function $\mathcal{F}$ is:

$$\mathcal{F} = \sum_{i=1}^{N} \log P(y_i) = -\frac{N}{2}(d\log(2\pi) + \log|C| + tr(C^{-1}S)) \tag{3.8}$$

where $C = WW^T + \sigma^2 I_d$ and S is the covariance matrix of Y, $S = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mu)(y_i - \mu)^T$.

There are two approaches to compute the model parameters. By taking derivative of $\mathcal{F}$ with respect to the model parameters, we can obtain the close-form solution of the optimal model parameter. We can also iteratively estimate the model parameters using

expectation-maximization (EM) algorithm [47]. Here we only cover the first approach. The detailed derivation of learning PPCA using EM algorithm can be found in [41].

For the derivative approach, computing μ is straightforward:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{3.9}$$

To compute W, the partial derivative of $\mathcal{F}$ with respect to W is,

$$\frac{\partial \mathcal{F}}{\partial W} = N(C^{-1}SC^{-1}W - C^{-1}W) \tag{3.10}$$

by setting equation (3.10) to be zero, we get:

$$SC^{-1}W = W \tag{3.11}$$

Denote the eigen-decomposition of S as $S = UDU^T$, where U is a $d \times d$ orthogonal matrix and D is a diagonal matrix whose diagonal elements contain the eigenvalues of S. Also, define the singular value decomposition [32] of W as $W = U_w L V_w^T$ where U_w is a $d \times q$ orthonormal matrix, L is a $q \times q$ diagonal matrix and V_w is a $q \times q$ orthogonal matrix. After manipulation of equations, it can be shown that:

$$W = U_q(D_q - \sigma^2 I_q)^{1/2}R \tag{3.12}$$

where U_q is a $d \times q$ matrix contains q eigenvectors of S that correspond to the q largest eigenvalues, which are contained in the $q \times q$ diagonal matrix D_q. R is an arbitrary $q \times q$ orthogonal matrix.

From equation (3.12), it is clear that PPCA spans the same subspace as PCA.

To compute σ^2, we maximize equation (3.14) with respect to σ^2, and it gives:

$$\sigma^2 = \frac{1}{d-q} \sum_{i=q+1}^{d} \lambda_i \tag{3.13}$$

where λ_i is the i-th largest eigenvalue in matrix S.

3.1.3 Analysis

Since all the inference and learning equations are well defined, we can now describe the geometrical properties of the PPCA model.

First, we will show the log likelihood $\log P(y)$ is determined by two distances. To simplify the analysis, we set μ to be zero so as to make the origin of the subspace become O.

From equation (3.4):

$$\log P(y) = -\frac{1}{2}(d \log(2\pi) + \log |C| + y^T C^{-1} y) \tag{3.14}$$

where $d \log(2\pi)$ and $\log |C|$ are two constants. Also, according to Sherman-Morrison matrix inverse formula [32], we can get:

$$C^{-1} = (WW^T + \sigma^2 I_d)^{-1} = \sigma^{-2} I_d - \sigma^{-2} W (W^T W + \sigma^2 I_q)^{-1} W^T \tag{3.15}$$

From equation (3.12) and equation (3.15) we can derive:

$$\begin{aligned} y^T C^{-1} y &= \sigma^{-2} y^T (I - U_q U_q^T) y + y^T U_q D_q^{-1} U_q^T y \\ &= \sigma^{-2} d_e + d_m \end{aligned} \tag{3.16}$$

where $d_e = y^T(I - U_q U_q^T)y$ is the Euclidean distance from y to the subspace spanned by U_q, and $d_m = y^T U_q D_q^{-1} U_q^T y$ is the Mahalanobis distance of the linear projection of y to O in this subspace.

According to this result, points with small Euclidean distances to the subspace are not guaranteed to have large likelihoods. They also have to be close to the center O. As a consequence, PPCA is a probabilistic model that covers only a local region in the subspace, as is shown in Figure 3.1.

Next, we will demonstrate the effect of prior on latent variables z. Let y_o be a point that lies on the subspace; that is, $y_o = U_q U_q^T y_o$. It can be shown that the reconstruction of y_o from its corresponding latent variables is:

$$E[P(y|E[P(z|y_0)])] = U_q(D_q - \sigma^2) D_q^{-1} U_q^T y_o \neq y_o \tag{3.17}$$

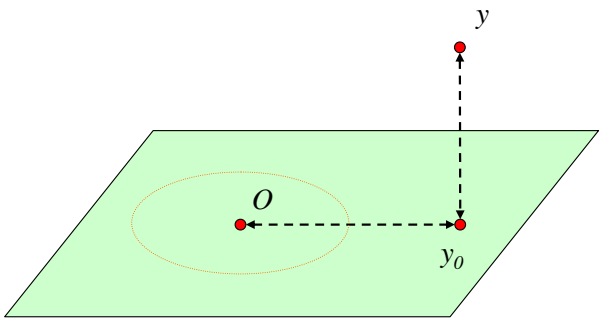

Figure 3.1 An example of using two distances to compute log-likelihood $\log P(y)$. Here y is a 3-dimensional vector, the green parallelogram represents the 2-D subspace spanned by U_q in PPCA model, and $y_0 = U_q y$ is the linear projection of y into this subspace. The orange ellipsoid whose shape and size are determined by D_q illustrates the effect of Mahalanobis distance. $\log P(y)$ is then determined by the Euclidean distance between y and y_0 and the Mahalanobis distance between y_o and O.

This is because the Gaussian prior $P(z)$ pulls latent variable toward the origin. As a result, the reconstruction of y_o is closer to O, too.

3.2 Mixtures of Probabilistic Principal Component Analysis

For data embedded on a nonlinear manifold, a single PPCA can not fit data well. However, since a nonlinear manifold can be regarded as a number of linear patches "glued" together, it can be modelled using a collection of PPCAs. This extension of PPCA model is termed mixtures of probabilistic principal component analyzers (MPPCA) [5].

3.2.1 Inference

A mixture of K probabilistic principal component analyzers has following parameters, $\Theta = (\Theta_1, \ldots . \Theta_K, \pi)$, where $\Theta_s = (\mu_s, W_s, \sigma_s^2)$ are parameters for the s-th PPCA model and $\pi = (\pi_1, \pi_2, \ldots, \pi_K)$ is the discrete prior probabilities for the K PPCA models.

Based on the parameters, the likelihood of $P(y)$ in a MPPCA is defined as:

$$P(y) = \sum_{s=1}^{K} P(y|s)P(s) \tag{3.18}$$

where s is a index variable with $P(s = i) = \pi_i$ and $P(y|s = i) = P(y|\Theta_i)$.

We can also infer s from y:

$$P(s|y) = \frac{P(y|s)P(s)}{\sum_{s=1}^{K} P(y|s)P(s)} \tag{3.19}$$

In addition, we can and z^s, the latent variables in the s-th PPCA model, given y and s. Since $P(z^s|y, s) = P(z^s|y, \Theta_s)$,

$$P(z^s|y, s) \sim \mathcal{N}(\mu_s, \Sigma_s) \tag{3.20}$$

in which (μ_s, Σ_s) are defined in (3.5).

In MPPCA, $P(y)$,the likelihood that sample y is generated from the model, is the most commonly used inference. $P(s|y)$ and $P(z^s|y, s)$ are mainly used during learning which will be described in the following section.

3.2.2 Learning

Learning MPPCA, like learning in PPCA, is based on maximum likelihood estimation, but because of the introduction of model index variable s it does not have a close-form

solution. Instead, the learning algorithm is an iteratively procedure based expectation maximization.

Given i.i.d. data set $Y = \{y_1, \ldots, y_N\}$, the objective is to find Θ that maximizes:

$$\mathcal{L} = \log P(Y|\Theta) = \sum_{i=1}^{N} \log P(y_i|\Theta) \tag{3.21}$$

According to Jensen's Inequality it can be shown that:

$$\mathcal{L} \geq \mathcal{F} = \sum_{i=1}^{N} \sum_{s_i=1}^{K} \int Q(s_i, z^{s_i}) \log \frac{P(y_i, s_i, z^{s_i})}{Q(s_i, z^{s_i})} dz^{s_i} \tag{3.22}$$

With $Q(s_i, z^{s_i})$ being an arbitrary distribution. However, if $Q(s_i, z^{s_i}) = P(s_i, z^{s_i}|y_i)$ the equality in equation (3.22) holds, i.e. $\mathcal{L} = \mathcal{F}$. Therefore, the EM algorithm for MPPCA works as follows.

Starting with the initial model parameters $\Theta^{(0)}$. In the E-step, compute $P(s_i, z^{s_i}|y_i, \Theta^{(0)})$. Since $P(s_i, z^{s_i}|y_i) = P(z^{s_i}|y_i, s_i)P(s_i|y_i)$, $P(s_i|y_i, \Theta^{(0)})$ and $P(z^{s_i}|y_i, s_i, \Theta^{(0)})$ are the terms actually computed.

In the M-step, by plugging the results of $P(s_i, z^{s_i}|y_i, \Theta^{(0)})$ into $Q(s_i, z^{s_i})$ in equation (3.22) and taking the derivative of $\mathcal{F}$ with respect to the model parameters, we can then obtain the following equations for model update:

$$\pi_k = \frac{1}{N} \sum_{i=1}^{N} P(s_i = k|y_i) \tag{3.23}$$

$$\mu_k = \frac{\sum_{i=1}^{N} P(s_i = k|y_i) y_i}{\sum_{i=1}^{N} P(s_i = k|y_i)} \tag{3.24}$$

$$S_k = \frac{\sum_{i=1}^{N} P(s_i = k|y_i)(y_i - \mu_j)(y_i - \mu_j)^T}{\sum_{i=1}^{N} P(s_i = k|y_i)} \tag{3.25}$$

After S_k is now known, W_k and σ_k^2 can be computed according to equation (3.12) and equation (3.13).

3.2.3 Analysis

As is pointed out in Section 3.2.1, the practical inference in the MPPCA model is rather limited. While $P(y)$ can describe the relationship between sample y and the underlying manifold, $P(s|y)$ and $P(z^s|y,s)$ provide little additional information.

This limitation is from the structure of MPPCA. Although MPPCA parameterizes a nonlinear manifold using a collection of local linear patches, it provides no information about how these linear patches are aligned in the space. Therefore, given two samples y_1 and y_2 lying on the manifold, even though the likelihoods $\{P(s|y_1), P(z^s|y_1,s), P(s|y_2), P(z^s|y_2,s)\}_s$ can be computed from the model, these measurements do not provide sufficient information to infer the geodesic relationship between y_1 and y_2.

In the next chapter, we will present the global coordination model, which is an extension of MPPCA. The global coordination model maps local latent variables z^S to a global coordinated system in which the geodesic relationships among samples are clearly defined. Before that, we will first introduce online updating PPCA and discriminative PPCA model.

3.3 Adaptive Probabilistic Principal Component Analysis

In the following two sections, we shift the theme back to a single probabilistic principal component analyzer, but address the learning problem in a dynamic setting. Unlike the discussion in section 3.1.2 where the training data set is fixed and static, here we assume the training set is dynamically expanding as new training data are continuously added in. As a result, the PPCA model has to be adaptively adjusted according to the new data.

This adaptive algorithm can be related to manifold learning. As has been pointed out in Chapter 2, every local patch on a smooth nonlinear manifold can be approximated using a PPCA. We envision a learning problem in which a process moves along the surface of a nonlinear manifold and contineously collects data samples on its way. The learning task is to adaptively adjust a PPCA model such that it always fits the local linear region on the manifold where the process is currently located. This is exactly the scenario of object tracking using an adaptive appearance model [48] [49] [50] [51], but here we will only deal with the abstract problem.

PPCA parameters include μ, which captures the mean of the data set, and matrix W, which according to (3.12) is determined by the eigenvectors and the eigenvalues of the data covariance matrix. These eigenvectors and the eigenvalues of are computed using singular value decomposition (SVD) [32]. Therefore, to incrementally update the PCA a dynamic data set requires an incremental SVD algorithm. Incremental singular value

decomposition is a well studied topic and has been successfully applied in text retrieval [42], image processing [43], and computer vision [44].

In the section, we start with an introduction of the incremental SVD algorithm. Based on which, we present two algorithms that adaptively updates PPCA model. If the dynamic data set has zero mean, i.e. μ is fixed to zero and only W has to be changed. Updating PPCA model is a straight forward application of the incremental SVD algorithm. This is approach that most adaptive PCA models are based on [42] [43] [44] [48]. However, the zero mean assumption is a strong constraint and is apparently violated in our adaptive manifold learning problem. A more general adaptive PCA model shall allow both μ and W to be dynamically changed. The is a difficult problem since when mean is varying the data covariance matrix has to be constantly adjusted according the the new mean, and there has not little work one this problem. Hall et. al. [52] propose a algorithm for this but it only gives an approximate results and only handles one new datum per update. In this section, we propose a novel adaptive PCA algorithm which handles multiple new data pre update and renders exact solutions. Our algorithm is also based on the incremental SVD algorithm, and it is has little overhead compared with the zero-mean adaptive PPCA algorithm. In addition, we show that by introducing a forgetting factor in our incremental learning framework, our algorithm is capable of learning a drifting concept. That is, the effect of data observed long time ago will be gradually reduced while updating the model.

3.3.1 Incremental Singular Value Decomposition

In this section, we introduce an incremental singular value decomposition algorithm called *Sequential Karhunen-Loeve algorithm* (SKL) based on the work by Levy et. al. [43]. This problem can defined as follows. Given two matrices, $A \in \mathcal{R}^{M \times L}$ and $B \in \mathcal{R}^{M \times K}$ with $M \geq (L + K)$, assuming the SVD of A are known and the task is to compute the SVD of matrix $[A|B]$ efficiently.

Let $A = UDV^T$ be the SVD result of A, where U is a $M \times L$ orthogonal matrix, D is a $L \times L$ diagonal matrix, and V is a $L \times L$ orthonormal matrix. Denote $[U|U_{new}]$ as the orthogonal bases of the subspace spanned by $[A|B]$. It can be shown that $U_{new}^T A = 0$ and $(UU^T + U_{new}U_{new}^T)B = B$. Based on these two properties, $[A|B]$ can be factorized as:

$$\begin{aligned} [A|B] &= U'D'V' \\ U' &= [U|U_{new}], \quad D' = \begin{bmatrix} D & U^T B \\ 0 & U_{new}^T B \end{bmatrix}, \quad V' = \begin{bmatrix} V^T & 0 \\ 0 & I \end{bmatrix}, \end{aligned} \tag{3.26}$$

Let the SVD of D' be $D' = \tilde{U}\tilde{D}\tilde{V}^T$ and the SVD of $[A|B]$ be $[A|B] = \hat{U}\hat{D}\hat{V}^T$. It can be shown that,

$$[A|B] = U'D'V' = (U'\tilde{U})\tilde{D}(\tilde{V}^T V'^T) = \hat{U}\hat{D}\hat{V}^T \tag{3.27}$$

Sequential Karhunen-Loeve algorithm

Given matrices U_K, D_K and B, in which U_K is a $M \times K$ matrix containing the K most significant left singular vectors of A, D_K is a $K \times K$ diagonal matrix with the the K most significant singular values of A in its diagonal entities and B is the new additional data matrix, the task is to return the K most significant left singular vectors and singular values of matrix $[A|B]$.

1. Compute the QR-decomposition [32] : $[U_K D_K|B] = QR$
2. Compute the SVD of R as the product of $R = \tilde{U}\tilde{D}\tilde{V}^T$
3. Reduce the diagonal matrix $\tilde{D}$ to a $K \times K$ matrix by keeping only the K most significant singular values in $\tilde{D}$. The size of $\tilde{D}$ can be further reduced by removing singular values that are negligibly small.
4. Reduce $\tilde{U}$ in accordance with $\tilde{D}$, such that the j-th column in $\tilde{U}$ contains the singular vector that corresponds to the singular value in $[\tilde{D}]_{j,j}$.
5. Set $U_K = Q * \tilde{U}$ and $D_K = \tilde{D}$.

Table 3.1 Details of the sequential Karhunen-Loeve algorithm

$$\text{where} \quad \hat{U} = U'\tilde{U}, \quad \hat{D} = D', \quad \hat{V} = V'\tilde{V}$$

That is, to compute $\hat{U}$, $\hat{D}$ and $\hat{V}$, we have to get U_{new}, and then then compute the SVD of D'.

Now if we modify the problem such that only the K most significant left singular vectors and singular values of matrix $[A|B]$ are needed, the solution is still based on (3.26) and (3.27) but it requires less computation. This algorithm is called Sequential Karhunen-Loeve algorithm (SLD) [43] and its detail is shown in Table 3.1. Note that since only the left singular vectors are needed, $\tilde{V}$ can be discarded without affecting the result.

3.3.2 Adaptive PPCA

Based on the SKL algorithm, we now present two algorithms to adaptively update the PPCA model. The first one works on a degenerative case in which the sample mean has to be fixed at zero. The second one is our proposed algorithm, which can be applied to general PPCA models without any additional constraints.

The adaptive PPCA problem can be formulated as follows. Let $Y = [y_1|y_2|\dots|y_M]$ be the original training data set, based on which the PPCA model parameters μ and W are computed according to (3.9) and (3.12) respectively. Let $Y^n = [y_{M+1}|y_{M+2}|\dots|y_{M+N}]$ be the new additional data added into the dataset. The task is to compute the PPCA model parameters μ_{new} and W_{new} with respect to the dataset $[Y|Y^n]$, using (μ, W) and Y_{new} only.

Starting with the zero-mean case, in which $\mu = \mu_{new} = 0$. Let $S_{new} = \frac{1}{M+N}\sum_{i=1}^{M+N}(y_i - \mu_{new})(y_i - \mu_{new})^T$ be the covariance matrix of $[Y|Y^n]$. S_{new} can be expressed in matrix form.

$$S_{new} = \frac{1}{M+N}[Y|Y^n][Y|Y^n]^T \tag{3.28}$$

According to (3.12), for a PPCA model with K-dimensional latent variables, only the K most significant eigenvalues and eigenvectors of S_{new} are needed, which can be obtain by computing the K most significant singular values and singular vectors of $[Y|Y^n]$. Now assuming that matrices U_K and D_K which contain the the K most significant left singular vectors and singular values of Y are known, this becomes exactly the same incremental

Zero-Mean Adaptive PPCA

1. *Initialization:* Let $Y = [y_1|\dots|y_M]$ be the initial data set. Calculate the SVD of $Y = UDV^T$ and keep only the K most most significant left singular vectors and singular values in U_K and D_K. Re-scale D_K to $D_K = \sqrt{\frac{1}{M}}D_K$. Then, use U_K and D_K to compute W according to (3.12).

2. *Update*: Given (W, U_K, D_K) and the newly observed data $Y^{(n)} = [y_1^{(n)}|\dots|y_N^{(n)}]$,

   (a) Set $D_K = \sqrt{M}D_K$. Compute U_K^{new} and D_K^{new} using the SKL-algorithm with U_K, D_K and $Y^{(n)}$ where $Y^{(n)}$ represents the new additional data matrix. Then, set $D_K^{new} = \sqrt{\frac{1}{N+M}}D_K^{new}$.

   (b) Compute W_{new} by applying U_K^{new} and D_K^{new} in (3.12).

   (c) Set $M = M + N$, $W = W_{new}$, $U_K = U_K^{new}$ and $D_K = D_K^{new}$

Table 3.2 Details of zero-mean adaptive PPCA algorithm

SVD problem descried in the previous section. Hence, the zero-mean PPCA model can be efficiently by directly applying the SKL algorithm. The details is shown in Table 3.2.

Note that in the adaptive PPCA model since only the K most significant eigenvalues of the covariance matrix S are kept, we can not no longer update σ^2 using (3.13). As a matter of fact, now we do not have sufficient information to compute the exact σ^2. A intuitive approximation is to set σ^2 as a fixed fraction of the sum of the K most significant eigenvalues:

$$\sigma^2 = l\sum_{i=1}^{K}[D_K]_{i,i}^2 \tag{3.29}$$

where $0 < l < 1$. In addition, we have to ensure that $\sigma^2 < [D_K]_{K,K}^2$.

If the sample mean also changes according to the new additional data, the PPCA update become more complicated since the mean variation affects the covariance matrix.

In the following, we provide the derivation of our algorithm to efficiently update a general PPCA model with zero-mean constraint. Starting with the mean update. According (3.9) it is straight forward to compute μ_{new} using μ and Y^n.

$$\begin{aligned}
\mu_{new} &= \frac{1}{(M+N)} \sum_{i=1}^{M+N} y_i = \frac{1}{(M+N)} \sum_{i=1}^{M} y_i + \frac{1}{(M+N)} \sum_{i=M+1}^{M+N} y_i \\
&= \frac{M}{(M+N)} \left(\frac{1}{M} \sum_{i=1}^{M} y_i \right) + \frac{N}{(M+N)} \left(\frac{1}{N} \sum_{i=M+1}^{M+N} y_i \right) \\
&= \frac{M}{(M+N)} \mu + \frac{N}{(M+N)} \mu_{add}
\end{aligned} \tag{3.30}$$

where $\mu_{add} = \frac{1}{N} \sum_{i=M+1}^{M+N} y_i$ is the mean of the additional data samples.

Computing W_{new} is more difficult. According to (3.12), W_{new} depends on the eigenvalues and the eigenvectors of the covariance matrix $S_{new} = \frac{1}{(N+M)} \sum_{i=1}^{N+M} (y_i - \mu_{new})(y_i - \mu_{new})^T$, but the original data set $Y = [y_1|y_2|\dots|y_M]$ is no longer available. We first decompose S_{new} as:

$$S_{new} = \frac{1}{N+M} \sum_{i=1}^{M} (y_i - \mu_{new})(y_i - \mu_{new})^T + \frac{1}{N+M} \sum_{i=M+1}^{N+M} (y_i - \mu_{new})(y_i - \mu_{new})^T \tag{3.31}$$

Let $S = \frac{1}{M} \sum_{i=1}^{M} (y_i - \mu)(y_i - \mu)^T$ be the covariance matrix of Y. It can be shown that,

$$\sum_{i=1}^{M} (y_i - \mu_{new})(y_i - \mu_{new})^T = \sum_{i=1}^{M} (y_i - \mu + \mu - \mu_{new})(y_i - \mu + \mu - \mu_{new})^T$$

$$
\begin{aligned}
&= \sum_{i=1}^{M}(y_i-\mu)(y_i-\mu)^T + M(\mu-\mu_{new})(\mu-\mu_{new})^T \\
&= MS + M(\mu-\mu_{new})(\mu-\mu_{new})^T \qquad (3.32)
\end{aligned}
$$

Similarly, let $S_{add} = \frac{1}{N}\sum_{i=M+1}^{M+N}(y_i-\mu_{add})(y_i-\mu_{add})^T$ be the covariance matrix of Y^n,

$$
\sum_{i=M+1}^{M+N}(y_i-\mu_{new})(y_i-\mu_{new})^T = NS_{add} + N(\mu_{add}-\mu_{new})(\mu_{add}-\mu_{new})^T \qquad (3.33)
$$

According to (3.30), $\mu-\mu_{new} = \frac{N}{M+N}(\mu-\mu_{add})$ and $\mu_{add}-\mu_{new} = \frac{M}{M+N}(\mu_{add}-\mu)$. Together with (3.31),(3.32) and (3.33), S_{new} can be factorized as:

$$
S_{new} = \frac{M}{(N+M)}S + \frac{N}{(N+M)}S_{add} + \frac{MN}{(N+M)^2}(\mu_{add}-\mu)(\mu_{add}-\mu)^T \qquad (3.34)
$$

That is, S_{new} can be computed using S, S_{add}, μ and μ_{add}.

Let $\bar{Y} = [y_1-\mu|\ldots|y_M-\mu]$ and $\bar{Y}^n = [y_{M+1}-\mu_{add}|\ldots|y_M-\mu_{add}]$. Equation (3.34) can be expressed in matrix form:

$$
S_{new} = \frac{1}{M+N}\left[\bar{Y}|\bar{Y}^n|\sqrt{\frac{MN}{M+N}}(\mu_{add}-\mu)\right]\left[\bar{Y}|\bar{Y}^n|\sqrt{\frac{MN}{M+N}}(\mu_{add}-\mu)\right]^T \qquad (3.35)
$$

Note that this result is very similar to (3.28), but here $\bar{Y}$ and $\bar{Y}^n$ are data blocks with their means subtracted. In addition, there is an additional column $\left[\sqrt{\frac{MN}{M+N}}(\mu_{add}-\mu)\right]$ to

account for the mean change in the covariance matrix. Based on (3.35), we present an adaptive learning algorithm for a general PPCA model shown in Table 3.3.

General Adaptive PPCA

1. *Initialization:* Let $Y = [y_1|\dots|y_M]$ be the initial data set.
   (a) Compute μ according to (3.9), and set $\bar{Y} = [y_1 - \mu|\dots|y_M - \mu]$.
   (b) Calculate the SVD of $\bar{Y} = UDV^T$ and keep only the K most most significant left singular vectors and singular values in U_K and D_K. Re-scale D_K to $D_K = \sqrt{\frac{1}{M}}D_K$. Then, use U_K and D_K to compute W according to (3.12).
2. *Update*: Given (μ, W, U_K, D_K) and the newly observed data $Y^{(n)} = [y_1^{(n)}|\dots|y_N^{(n)}]$,
   (a) Set $\mu_{add} = \frac{1}{N}\sum_{j=1}^{N} y_j^{(n)}$ and $\bar{Y}^{(n)} = [y_1^{(n)} - \mu_{add}|\dots|y_N^{(n)} - \mu_{add}]$. Then, compute μ_{new} using (3.30).
   (b) Set $D_K = \sqrt{M}D_K$. Compute U_K^{new} and D_K^{new} using the SKL-algorithm with U_K, D_K and $[Y^{(n)}|\frac{MN}{(M+N)}(\mu_{add} - \mu)]$ where $[Y^{(n)}|\frac{MN}{(M+N)}(\mu_{add} - \mu)]$ represents the new additional data matrix. Then, set $D_K^{new} = \sqrt{\frac{1}{N+M}}D_K^{new}$.
   (c) Compute W_{new} by applying U_K^{new} and D_K^{new} in (3.12).
   (d) Set $M = M + N$, $\mu = \mu_{new}$, $W = W_{new}$, $U_K = U_K^{new}$ and $D_K = D_K^{new}$

Table 3.3 Details of our adaptive PPCA learning algorithm

Adaptively updating the mean and the covariance matrix is a difficult problem which has seldom been addressed before. Most work on adaptive PCA algorithms assume μ is fixed to zero [42] [20][43][48]. Hall et. al. [52] present an adaptive PCA algorithm without zero mean constraint but it is a rough approximation. Here we present an general adaptive PPCA algorithm that provides the exact solution. Our algorithm is close to the zero-mean adaptive PCA algorithm shown in Table 3.2. As a matter of fact, by neglecting the constant time mean update, our algorithm only expands the new data

matrix by one new column $\left[\sqrt{\frac{MN}{M+N}}(\mu_{add} - \mu)\right]$ while running the SKL algorithm, so its computational complexity is the same as the zero-mean algorithm.

3.3.3 Forgetting Factor

In the discussion above, we assume that the dynamic data set are generated from a fixed latent variable model. Therefore, after data set has grown to a certain size, the PPCA model parameters converge and the additional new data will have little effect. This is shown in Table 3.3 step 2(b). As $M \gg N$, $U_K D_K$ will dominate and new data matrix $\bar{Y}^n$ will become negligible while computing U_k^{new} and D_K^{new}.

Now considering a different setting where data are generated from a dynamic latent variable model whose model parameters slowly change over time. The goal is to recover the latest model parameters from the dynamic data set. Since only the recent data samples in the data set are generated from the latest model parameters, we have to remove old samples from the data set before computing the model parameters. This procedure is trivial if all the data samples are actually saved, but since we only keep the statics of the data set, i.e. the mean and the covariance matrix, we need an efficient method to remove the effect of old data samples in these statistics. Here we shown that, this problem can be efficiently solved by introducing a forgetting factor.

Let $Y^{(t)} = [y_1^{(t)}| \ldots |y_N^{(t)}]$ be the new data block added into the data set at time t, and $\mu^{(t)} = \frac{1}{N}\sum_{i=1}^{N} y_i^{(t)}$ is its mean. Starting with initial data block $Y^{(0)}$, the complete data set at time t is $Y_t = [Y^{(0)}|Y^{(1)}| \ldots |Y^{(t)}]$. Now we create a weighted data set Y_t^w by assigning a weight to each data block using a forgetting factor f, $0 < f < 1$, suck that the weight

for block $Y^{(i)}$ is f^{t-i}. That is, $Y_t^w = [f^t Y^{(0)} | f^{t-1} Y^{(1)} | \dots | Y^{(t)}]$. Denote μ_t^w and S_t^w as the mean and covariance matrix of Y_t^w, it can be shown that

$$\mu_t^w = \frac{\sum_{i=0}^t f^{t-i}\mu^{(i)}}{\sum_{i=0}^t f^{t-i}}, \quad S_t^w = \frac{\sum_{i=0}^t f^{t-i}[\bar{Y}^{(i)}][\bar{Y}^{(i)}]^T}{N\sum_{i=0}^t f^{t-i}} \tag{3.36}$$

Since $0 < f < 1$, the weights of these blocks exponentially decrease in reverse chronicle order. Blocks added into the data set long time ago will receive weights close to zero. Therefore, the weighted mean μ_t^w and covariance matrix S_t^w are mainly determined by the most recent data blocks.

Let $F_t = \sum_{i=0}^t f^{t-i}N$, F_t can be expressed as a recursive equation,

$$M_t = \sum_{i=0}^{t-1} f^{t-i}N + N = fM_{t-1} + N \tag{3.37}$$

Similarly, μ_t^w and S_t^w can be expressed as two recursive equations:

$$\mu_t^w = \left(\frac{N}{fM_{t-1}+N}\right)\mu^{(t)} + \left(\frac{fM_{t-1}}{fM_{t-1}+N}\right)\mu_{t-1}^w \tag{3.38}$$

$$S_t^w = \left(\frac{N}{fM_{t-1}+N}\right)\frac{[\bar{Y}^{(t)}][\bar{Y}^{(t)}]^T}{N} + \left(\frac{fM_{t-1}}{fM_{t-1}+N}\right)S_{t-1}^w \tag{3.39}$$

Based on result in (3.37),(3.38) and (3.39), we can then modify our adaptive PCA algorithm to incrementally incrementally update the PPCA model with weighted data set. The modification is little: we simply attach the forgetting factor f to M. The detailed modified algorithm is shown in Table 3.4. Note that M_t will converges to $\frac{N}{1-f}$ as

Weighted General Adaptive PPCA

1. *Initialization:* Let $Y = [y_1|\dots|y_M]$ be the initial data set.
   (a) Compute μ according to (3.9), and set $\bar{Y} = [y_1 - \mu|\dots|y_M - \mu]$.
   (b) Calculate the SVD of $\bar{Y} = UDV^T$ and keep only the K most most significant left singular vectors and singular values in U_K and D_K. Re-scale D_K to $D_K = \sqrt{\frac{1}{M}}D_K$. Then, use U_K and D_K to compute W according to (3.12).
2. *Update*: Given (μ, W, U_K, D_K) and the newly observed data $Y^{(n)} = [y_1^{(n)}|\dots|y_N^{(n)}]$,
   (a) Set $\mu_{add} = \frac{1}{N}\sum_{j=1}^{N} y_j^{(n)}$ and $\bar{Y}^{(n)} = [y_1^{(n)} - \mu_{add}|\dots|y_N^{(n)} - \mu_{add}]$. Then, compute μ_{new} using (3.30).
   (b) Set $D_K = \sqrt{M}D_K$. Compute U_K^{new} and D_K^{new} using the SKL-algorithm with U_K, D_K and $[Y^{(n)}|\frac{fMN}{(fM+N)}(\mu_{add} - \mu)]$ where $[Y^{(n)}|\frac{fMN}{(fM+N)}(\mu_{add} - \mu)]$ represents the new additional data matrix. Then, set $D_K^{new} = \sqrt{\frac{1}{N+fM}}D_K^{new}$.
   (c) Compute W_{new} by applying U_K^{new} and D_K^{new} in (3.12).
   (d) Set $M = fM + N$, $\mu = \mu_{new}$, $W = W_{new}$, $U_K = U_K^{new}$ and $D_K = D_K^{new}$

Table 3.4 Details of our weighted adaptive PPCA learning algorithm

t increases to infinity. Therefore, $\frac{N}{fM_{t-1}+N} \geq \frac{1-f}{f}$. The statistics of new data block always receive a certain weight while updating the weighted mean and covariance matrix.

3.4 Discriminative Probabilistic Principal Component Analyzer

In the section, we propose a novel algorithm to perform discriminative analysis on a generative model. We first present the base algorithm. Then we expand the model to become an online learning algorithm. As will be shown, this adaptive discriminative algorithm is an extension of the adaptive PPCA model described in the previous section.

3.4.1 Discriminative PPCA

We envision a object-background classification problem. Given a test data set ,$Y = \{y_1, \ldots, y_L\}$, $y_i \in \mathcal{R}^d$, whose samples are generated either from the objected class or the background class. The task is to select the most likely sample in Y that comes from the object class. We use two sets of training examples, $Y^o = \{y_1^o, \ldots, y_M^o\}$ contains examples belong to the object class and $Y^b = \{y_1^b, \ldots, y_N^b\}$ contains the background examples. Data in the training and testing sets are sampled according to the same distribution. For the object class, we know it can be modeled by a PPCA, and we train a the model using Y^o. As for the background class, it is difficult to build any model for it. If not background training examples are available, we can pick out the most likely sample y_k using

$$k = \arg\max_i P(y_i|\Theta) \tag{3.40}$$

where $\Theta = \{\mu, W, \sigma^2\}$ are the PPCA model parameters trained using Y^o. With background data set Y^b available, we want to use Y_b to tune PPCA so as to improve the selection performance.

Let V be a $q \times d, (q < d)$ matrix, which linearly projects y to a subspace. It can be shown that in this projected subspace the likelihood in (3.4) becomes

$$P(Vy|\Theta, V) \sim \mathcal{N}(V\mu, VCV^T), \quad C = WW^Y + \sigma^2 I_d \tag{3.41}$$

Given background training set $Y^b = \{y_1^b, \ldots, y_M^b\}$, we want to find the linear projection V^* that *minimizes* the log likelihood:

$$V^* = \arg\min_V \sum_{i=1}^{N} \log P(Vy_i^b|\Theta, V). \tag{3.42}$$

That is, we are projecting the data and the PPCA model to a subspace in which the likelihood $P(Vy|\Theta, V)$ of the background samples are suppressed. Therefore, preforming the selection process in this subspace is less likely to make error:

$$k = \arg\max_i P(V^* y_i|\Theta, V^*) \tag{3.43}$$

We now show how to compute V^*. According to (3.41),

$$\begin{aligned} \sum_{i=1}^{N} \log P(Vy_i^b|\Theta, V) &= \sum_{i=1}^{N} -\frac{1}{2}\Big(q\ln(2\pi) + \ln|VCV^T| + (y_i^b - \mu)^T V^T (VCV^T)^{-1} V(y_i^b - \mu)\Big) \\ &= -\frac{N}{2}\Big(q\ln(2\pi) + \ln|VCV^T| + trace\left((VCV^T)^{-1}(VS_W V^T)\right)\Big) \end{aligned} \tag{3.44}$$

where $S_W = \frac{1}{N}\sum_{i=1}^{N}(y_i^b - \mu)(y_i^b - \mu)^T$. By imposing the constraint $V^T C V = I_q$, it can be shown that,

$$V^* = \arg\max_{VCV^T = I_q} trace\left((VCV^T)^{-1}(VS_W V^T)\right) = \arg\max_V \frac{|VS_W V^T|}{|VCV^T|} \tag{3.45}$$

V^* can be obtained by solving the generalized eigenvalue problem of S_w and C. Let $\{v_1, \ldots, v_q\}$ be the q most significant generalized eigenvectors of S_w and C, $V^* = [v_1|\ldots|v_q]^T$.

3.4.2 Online Learning

The discriminative PPCA can extended to become an adaptive model. Assuming now the training data sets for both the object and the background are dynamically expanding, here we present an online algorithm to adaptively adjust projection matrix V^*.

We first show that S_w can also be efficiently computed. Denote $\mu_b = \frac{1}{N}\sum_{i=1}^{N} y_i^b$ is the mean and $S_b = \frac{1}{N}\sum_{i=1}^{N}(y_i^b - \mu_b)(y_i^b - \mu_b)^T$ as the covariance matrix of the background data. According to (3.32),

$$S_w = \frac{1}{N}\sum_{i=1}^{N}(y_i^b - \mu)(y_i^b - \mu)^T = S_b + (\mu_b - \mu)(\mu_b - \mu)^T \tag{3.46}$$

Given the dynamic data set Y_b, (μ_b, S_b) can be updated using our adaptive PPCA algorithm. To reduce the computation, we can approximate S_b by using its k most significant eigenvalues and eigenvectors, $S_b \approx U_k^b D_k^b {U_k^b}^T$, where U_k^b is a $d \times k$ orthogonal matrix and D_k^b is a $k \times k$ diagonal matrix. Updating (μ, C) is also based on our adaptive PPCA algorithm, since it is exactly the same adaptive learning problem described in Section 3.3.2.

With S_w and C are updated, we can then revise V^* by solving the generalized eigenvalue problem of S_w and C. As is detailed in [32], if $S_w = A^T A$ and $C = B^T B$, V^* can be computed by first performing a QR-decomposition:

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} Q_A \\ Q_B \end{bmatrix} R \tag{3.47}$$

where $A = Q_A R$ and $B = Q_B R$. Denote the singular value decomposition of Q_A as $Q_A = U_A D_A V_A^T$. V^* is determined by R and V_A,

$$V^* = R^{-1} V_A \tag{3.48}$$

Since S_w and C can be factorized as,

$$S_w = Sb + (\mu_b - \mu)(\mu_b - \mu)^T \approx [U_k^b D_k^{b\frac{1}{2}} | (\mu_b - \mu)][U_k^b D_k^{b\frac{1}{2}} | (\mu_b - \mu)]^T \tag{3.49}$$

$$C = WW^T + \sigma^2 I_d = [W|\sigma I_d][W|\sigma I_d]^T \tag{3.50}$$

We then set $A = [U_k^b D_k^{b\frac{1}{2}} | (\mu_b - \mu)]^T$ and $B = [W|\sigma I_d]$. Note that while computing S_w and C, we can also introduce the forgetting factor described in Sec 3.3.3 to down-weighted the data blocks that were added to the data set long time ago. We summarize this online discriminative algorithm in Table 3.5.

Online Discriminative PPCA

Given current model parameters (μ, W) and (μ_b, U_k^b, D_k^b) and new additional data set Y_n^o and Y_n^b:

1. Update (μ, W) using the weighted adaptive PPCA algorithm with data set Y_n^o.
2. Update (μ_b, U_k^b, D_k^b) using the weighted adaptive PPCA algorithm with data set Y_n^b.
3. set $A = [U_k^b D_k^{b\frac{1}{2}} | (\mu_b - \mu)]^T$ and $B = [W | \sigma I_d]$.
4. Compute Q_A and R according to (3.47).
5. Compute the SVD, $Q_A = U_A D_A V_A^T$.
6. Update V^* according to (3.48). That is, $V^* = R^{-1} V_A$.

Table 3.5 Details of our online discriminative PPCA algorithm

3.4.3 Multi-class Fisher's Discriminant Analysis

In this section, we point out the relation of our discriminative PPCA model to the traditional Fisher's discriminant analysis (FLD) [46]. The derivation here summarizes our earlier work in [53] and [54].

We first review the Fisher's discriminant analysis on the multiple-class problem. Denote $Y^i = \{y_1^i, \ldots, y_{N_i}^i\}$ as the training samples for class i, and assume there are K such classes. Let $\mu_k = \frac{1}{N_k} \sum_{i=1}^{N_k} y_i^k$ be the class sample mean for class i and $\mu = (\sum_{k=1}^K N_k \mu_k)/(\sum_{k=1}^K N_k)$ be the total sample mean. The FLD defines the within-class scatter matrix S_W and the between-class matrix S_B as:

$$S_W = \sum_{k=1}^{K} \sum_{i=1}^{N_k} (y_i^k - \mu_k)(y_i^k - \mu_k)^T, \quad S_B = \sum_{k=1}^{K} N_k (\mu_k - \mu)(\mu_k - \mu)^T \tag{3.51}$$

The multi-class FLD search for a matrix W to project samples to a subspace such that the determinant of the within-class scatter matrix is maximized while the determinant of the between-class scatter matrix is minimized. That is, the FLD finds matrix W that maximizes J:

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|} \tag{3.52}$$

That is,if the samples of each class form a cluster, W projects samples to a subspace in which the clusters are mostly separated from one another.

Back to the object-background classification problem, we have training examples $Y^o = \{y_1^o, \ldots, y_M^o\}$ for the object class and $Y^b = \{y_1^b, \ldots, y_N^n\}$ for the background class in which the sample distribution of the object class is known to be a Gaussian but the distribution of the background class is unknown.

Since we do not know how background samples can be clustered, we adopt a radical approach by treating each background sample as a single cluster. That is, now we have a multi-class classification problem with one object class and N background classes, and multi-class FDL projects samples to a subspace in which the object class are mostly separated from all the background class. Under the new setting, the within-class and between-class scatter matrices become,

$$S_W = \sum_{i=1}^{M}(y_i^o - \mu^o)(y_i^o - \mu^o)^T + \sum_{k=1}^{N}(y_k^b - \mu_k^b)(y_k^b - \mu_k^b)^T$$

$$= \sum_{i=1}^{M}(y_i^o - \mu^o)(y_i^o - \mu^o)^T \tag{3.53}$$

$$\begin{aligned} S_B &= M(\mu^o - \mu)(\mu^o - \mu)^T + \sum_{k=1}^{N}(\mu_k^b - \mu)(\mu_k^b - \mu)^T \\ &= M(\mu^o - \mu)(\mu^o - \mu)^T + N(\mu^b - \mu)(\mu^b - \mu)^T + NS_b \\ &= \frac{MN}{M+N}(\mu^o - \mu^b)(\mu^o - \mu^b)^T + NS_b \end{aligned} \tag{3.54}$$

where $\mu_k^b = y_k^b$, $\mu^b = \frac{1}{N}\sum_{i=1}^{N} y_i^b$, $\mu = (M\mu^o + N\mu^b)/(M+N)$ and $S_b = \frac{1}{N}\sum_{i=1}^{N}(y_i^b - \mu^b)(y_i^b - \mu^b)^T$. Replacing S_B and S_B in (3.51) with the results in (3.53) and (3.54), we get:

$$J(W) = \frac{|W^T(S_b + \frac{1}{M+N}(\mu^o - \mu^b)(\mu^o - \mu^b)^T)W|}{|W^T C W|} \tag{3.55}$$

with C being the covariance matrix of Y^o and S_b being the covariance matrix of Y^b. This is very similar to the result of our discriminative model shown in (3.42) and (3.46).

3.5 Summary

In this chapter, we being by reviewing the probabilistic principal component analyzer (PPCA) and the mixtures of probabilistic principal component analyzers (MPPCA). We describe their inference and learning algorithms and explain how these model can be related to manifold learning. MPPCA can only provide limited inference about the manifold. In the next chapter, we extend MPPCA to become a global coordination model (GC) which provides all sorts of manifold inferences.

In the second half of this chapter, we present the online learning algorithms for a single PPCA model. We first propose a novel online learning algorithm for PPCA. In contrast to the existing methods which assume the sample mean in PPCA is fixed, our method updates the mean and eigenbasis accurately and efficiently. We then present our algorithm to perform discriminative analysis on a PPCA to improve the robustness in a object-background classification problem. We further provide the online learning algorithm of this discriminative PPCA model. We will point out the relation of these online learning with our manifold learning algorithm in Chapter 7.

CHAPTER 4

Global Coordination of Local Linear Models

Global coordination model [19][20][21][6] is a manifold learning algorithm which extends mixtures of probabilistic principal component analyzers (MPPCA). It aligns latent variables in MPPCA in a global coordinate system, and uses this alignment model to provide nonlinear mapping from high-dimensional sample space to a single coherent low-dimensional vector space. With this mapping, the pairwise manifold distances between observation samples can be measured in low-dimensional global coordinate system.

Learning global coordination model is a ill-posed problem, because the low dimensional global coordinates of the training samples are unobserved. Learning the model requires imposing constraints on these hidden global coordinates. Wang et. al. [6] apply ISOMAP algorithm [13] to obtain a set of low-dimensional global coordinates and then formulate the learning problem as a regression problem. Teh et. al. [21] divide the learning problem in two stages. At the first stage, the mixture model is learned from the observed samples. After the mixture model is known, alignment parameters can be computed by solving an convex optimization problem formulated according to locally linear constraints on samples. Unlike the previous two approaches, Roweis et. al. [19] learn the parameters of the mixture model and the alignment simultaneously. They introduce

a regularizing term to specify the alignment constraints, and learn the model using the variational method [40].

As is pointed out in Sec 2.3.4, it is difficult to incorporate temporal constrains into ISOMAP and LLE algorithms. Because the theme of our work is to learn global coordination model through time series, in this chapter we will focus on Roweis's method. Roweis's algorithm is sensitive to initialization and has serious local minimum problem. In this next chapter, we will present our learning algorithm that is based on the same framework, but it is capable of learning from sequences. Experiments demonstrates that our model achieves superior learning results.

4.1 The Global Coordination Model

Global coordination model parameterizes is a mixture of principal component analyzers, but it further aligns the latent variables in the mixture model in a global coordinate system. With this additional alignment, global coordination model can map high-dimensional data to globally coordinated low-dimensional vectors, which we refer as *global coordinates* in the following. Global coordination model is a manifold learning algorithm. For high dimensional samples embedding on a low dimensional manifold, the goal of global coordination model is to learn the mapping from samples to low dimensional global coordinates in which the geodesic relationships among samples are preserved.

Following the notions uses in MPPCA in Sec 3.2.2, for a mixture of K PPCA models, denote $y \in \mathcal{R}^d$ the observed data, s the index of the selected PPCA model, and $z^s \in \mathcal{R}^q$

the latent variables of the s-th PPCA model. The joint probability of these parameters is:

$$P(y, z_s, s) = P(y|z^s, s)P(z_s|s)P(s) \tag{4.1}$$

in which $P(s)$ is the prior probability of local model s, $P(z^s|s)$ is a zero mean univariate Gaussian, i.e., $P(z^s|s) = \mathcal{N}(0, I_q)$, and $P(y|z^s, s)$ is defined as:

$$P(y|z^s, s) = \frac{1}{\sqrt{(2\pi)^d|\Psi|}} \exp(-\frac{1}{2}(y - \Lambda_s z^s - \mu_s)^T(\Psi)^{-1}(y - \Lambda_s z^s - \mu_s)) \tag{4.2}$$

In MPPCA, for each given s, the latent variable z^s is defined within the local coordinate system. The global coordination model transforms these latent variables, $\{z^s\}_{s=1}^K$, to a global coordinate system. Let g denote the global coordinate of data y that is generated from s-th local linear model with z^s, the transformation is defined by

$$g(s, z^s) = A_s z^s + \kappa_s, \tag{4.3}$$

where A_s is a full ranked matrix to ensure a invertible mapping, and κ_s is an offset. Since the mapping does not contain noise, $P(g|s, z^s)$ is a delta function:

$$P(g|s, z^s) = \delta(g - A_s z^s - \kappa_s) \tag{4.4}$$

Now, by including global coordinates g, the joint probability $P(y, g, z_s, s)$ is:

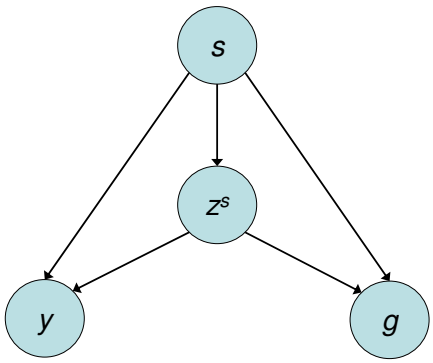

Figure 4.1 The global coordination model. The subgraph based on nodes s, z^s and g represent the MPPCA model. While the subgraph with nodes s, z^s and g describes the alignment of latent variables.

$$P(y, g, z_s, s) = P(y|z^s, s)P(g|z^s, s)P(z_s|s)P(s) \tag{4.5}$$

The relationships between variables y, g, z^s and s can be shown using the graphical model depicted in Figure 4.1.

4.1.1 Nonlinear Mapping

Given this model, we can perform statistical inference on observed data y and global coordinates g to compute the mapping between them. Firstly, the conditional probability $P(g|y)$ can be factorized as:

$$P(g|y) = \sum_{s=1}^{K} P(g|y, s)P(s|y) \tag{4.6}$$

where

$$P(g|y, s) = \int P(g|s, z_s)P(z_s|s, y)dz_s \tag{4.7}$$

$P(z^s|s,y)$ in (4.7)is defined in (3.20). Together with (4.4), it can be shown that $P(g|y,s)$ is a Gaussian:

$$\begin{aligned} P(g|y,s) &\sim \mathcal{N}(\kappa_s + A_s\mu_z, A_s^{-T}\Sigma_z^{-1}A_s^{-1}) \\ \mu_z &= \Sigma_z\Lambda_s^T\Psi^{-1}(y-\mu_s) \\ \Sigma_z &= (\Lambda_s^T\Psi^{-1}\Lambda_s + I_d)^{-1} \end{aligned} \tag{4.8}$$

Similarly, conditional probability $P(y|g)$ can be computed according to:

$$P(y|g) = \sum_{s=1}^{K} P(y|g,s)P(s|g). \tag{4.9}$$

with $P(y|g,s)$ also being a Gaussian.

$$P(y|g,s) \sim \mathcal{N}(\Lambda_s A_s^{-1}(g-\kappa_s) + \mu_s, \Psi) \tag{4.10}$$

$P(g|y)$ and $P(y|g)$ define the mapping between observed data sample y and its global coordinates g. Ideally, this mapping should be one-to-one; i.e. for every high-dimensional data sample lying on the manifold, it has its unique global coordinates, and vice versa. After including Gaussian noise during the mapping, the distributions of $P(g|y)$ and $P(g|y)$ are expected be unimodel.

According to (4.6), $P(g|y)$ is a mixture of K Gaussians weighted according $\{P(s|y)\}_{s=1}^{K}$. This does not contradict to the assumption above, since for most samples in the MP-

PCA model the probability mass function, $\{P(s|y)\}_{s=1}^{K}$, is a impluse functions peaks at certain s. That is, for a given y, $P(g|y) \sim P(g|y,s)$ for certain s. To ensure the actual distribution of $P(g|y)$ is unimodal through all possible y, there are constraints imposed in the learning process which will be explained in Section 4.3.

Because $P(g|y)$ and $P(y|g)$ are unimodals, they can be approximated as a single Gaussian with two parameters, the mean and the covariance matrix. For deterministic mappings between y and g, $E[P(g|y)]$ and $E[P(y|g)]$ are used in this work.

4.2 Learning from Embedding

Learning global coordination model is to estimate model parameters $\Theta_{gc} = \{\Lambda_s, \mu_s, A_s, \kappa_s, \pi_s, \Psi_s\}_s$ from a given set of observed $Y = \{y_1, y_2, \ldots, y_N\}$. Θ_{gc} can be divided into two sets, Θ_{mppca} and Θ_{align}, in which Θ_{mppca} comprises MPPCA parameters $\{\Lambda_s, \mu_s, \pi_s, \Psi_s\}_s$, and Θ_{gc} contains alignment parameters $\{A_s, \kappa_s\}_s$. Although global coordination model is an extension of MPPCA, learning Θ_{gc} is an ill-posed problem because of the additional alignment parameters, Θ_{gc}. This can explained using Figure 4.1. If parameters Θ_{gc} are estimated directly through maximum-likelihood estimation, since log-likelihood $\log P(Y)$ is solely determined by latent variables (s, z^s) and model parameters Θ_{mppca} alignment parameters Θ_{align} can be arbitrarily chosen without affecting $\log P(Y)$. Consequently, additional constraints have to be imposed during learning to ensure that the learned Θ_{align} maps latent variable z^s to the global coordinates g, which capture the intrinsic structures of the manifold.

These additional constraints are derived according to the properties of manifold. In this chapter, we focus on learning global coordination model based on spatial constraints on the manifold. We review three algorithms representing three different approaches to the learning problem: the regression approach, the post-alignment approach and the EM approach. The first two approaches which are based on the embedding results of ISOMAP or LLE are explained in this section. Learning via EM is a completely different algorithm, so it is left to the next section.

Our analysis on the advantages and limitations of these algorithms indicates that the learning results could be further improved by incorporating additional temporal constraints. This leads to the development of our algorithm that learns from sequences of observations, and the detailed of our algorithm will be covered in the next chapter.

4.2.1 The Regression Approach

The major difficulty in learning global coordination is that global coordinates, $G = \{g_1, g_2, \ldots, g_N\}$ with g_i being the global coordinates of observed sample y_i, are unobservable. If both Y and G are known, the learning problem is reduced to a regression problem. Based on this idea, Wang et. al. [6] apply ISOMAP to Y to obtain a set of low dimensional intrinsic parameters. They set these intrinsic parameters to be G and run the k-mean algorithm to divide samples into a number of clusters. Let s_i be the cluster index variable for g_i, the augment training set now becomes $\{(y_i, g_i, s_i)\}_{i=1}^{N}$. With all the latent variables in Figure 4.2 known, the model parameters can computed based on (4.8) or (4.10). Table 4.1 shows the detailed this learning algorithm.

Wang's algorithm

1. Apply ISOMAP on $Y = \{y_1, \ldots, y_N\}$ to obtain a set of local dimensional global coordinated intrinsic parameters, and use these as the global coordinates, $G = \{g_1, \ldots, g_N\}$, that correspond to Y.

2. Apply K-mean clustering algorithm on G to divide samples into clusters and obtain $S = \{s_1, \ldots, s_N\}$, where s_i is the cluster index variable for the i-th sample

3. Use the augmented training set $\{Y, G, S) = \{(y_i, g_i, s_i)\}_{i=1}^N$ to compute model parameters Θ_{gc}:

   (a) Use $\{s_i\}_{i=1}^N$ to compute model parameters π_s.

   $$\pi_k = \frac{\sum_{\{i|s_i=k\}} 1}{N}$$

   (b) Use $\{g_i, s_i)\}_{i=1}^N$ and (4.19) to computer alignment parameters $\{A_s, \kappa_s\}_s$.

   $$\kappa_k = \frac{\sum_{\{i|s_i=k\}} g_i}{\sum_{\{i|s_i=k\}} 1} \ , \quad A_k^T A_k = \pi_k \sum_{\{i|s_i=k\}} (g_i - \kappa_k)(g_i - \kappa_k)^T$$

   (c) Use $\{(y_i, g_i, s_i)\}_{i=1}^N$ and (4.10) to compute parameters $\{(\Lambda_s, \mu_s, \Psi_s)\}_s$

   $$\mu_k = \frac{\sum_{\{i|s_i=k\}} y_i}{\sum_{\{i|s_i=k\}} 1} \ , \quad \lambda_s = \pi_k \sum_{\{i|s_i=k\}} (y_i - \mu_k)(g_i - \kappa_k)^T A_k^{-T}$$

Table 4.1 The details of Wang's algorithm for learning global coordination based on the regression approach

Wang's method is built on top of ISOMAP, so the ISOMAP result becomes the bottleneck of this learning algorithm. Computing ISOMAP requires the pairwise geodesic distances among training samples which are estimated according to the Euclidean distances of samples that are nearest neighbors to each other. If training data are not densely sampled, the estimation of geodesic distances can be inaccurate which leads to inferior mapping result.

There is another questionable approach in Wang's algorithm. In step 2, k-mean clustering algorithm is applied on G not on Y. Hence, there is no guarantee that samples clustered into the same group their y-values in the high-dimensional space will be embedded on a low-dimensional linear subspace. If this property does not hold, the assumption of MPPCA model is violated and the learned global coordination model will have bad generalization result. A more reasonable approach will be to perform clustering using MPPCA algorithm on Y to ensure nonlinear manifold in the high dimensional space are divided into a number of locally linear regions. This modification makes Wang's algorithm closely related to the algorithm described in Section 4.2.2.

4.2.2 The Post-Alignment Approach

The global coordination model is a MPPCA model with additional alignment on the latent variables. A solution to this learning problem is to divide it into two subproblems and solve each separately. Based on this idea, Teh et. al. [21] first learn the MPPCA model from training set Y, then they learn the alignment parameters using their proposed coordination algorithm.

Learning the MPPCA model has been described in Section 3.2.2. After Θ_{mppca} is known, for a given sample, y_i, $P(s|y_i)$ and $P(z^s|y_i)$ can be inferred from the model. Let $v_i^s = P(s|y_i)$ and $\bar{z}_i^s = E[P(z^s|y_i, s)]$. According to Section 4.1.1, the global coordinates of y_i is defined as $E[P(g|y_i)]$. From (4.4), (4.7) and (4.6),

$$E[P(g|y_i)] = \sum_{s=1}^{K} E[P(g|s, y_i)]P(s|y_i) = \sum_{s=1}^{K} (A_s E[P(z^s|s, y_i)] + \kappa_s)v_i^s = \sum_{s=1}^{K} (A_s \bar{z}_i^s + \kappa_s)v_i^s \tag{4.11}$$

Denote L as a $d \times K(q+1)$ matrix $L = (A_1, \kappa_1, A_2, \kappa_2, \ldots, A_K, \kappa_K)$ and define u_i as a $K(q+1) \times 1$ vector:

$$u_i = \left(v_i^1 (z_i^1)^T, v_i^1, v_i^2 (z_i^2)^T, v_i^2, \ldots, v_i^K (z_i^K)^T, v_i^K\right)^T. \tag{4.12}$$

It can be shown that

$$E[P(g|y_i)] = Lu_i. \tag{4.13}$$

Given training set $Y = \{y_1, \ldots, y_N\}$ and MPPCA model Θ_{mppca}, we can compute matrix $U = (u_1, \ldots, u_N)$ and the global coordinates of sample y_i is in the i-th column in matrix LU.

Learning the alignment parameters is to compute matrix L such that LU represent a set of meaningful global coordinates of Y. In Teh et. al. [21], L is adjusted such that the global coordinates in LU will be close to the locally linear embedding (LLE) result on Y. Hence, their algorithm is termed *locally linear coordination* (LLC).

As is explained in Section 2.3.2, given a high-dimensional data set Y, the LLE algorithm first compute matrix W by minimizing the objective function:

$$\mathcal{E}(W) = \sum_i |y_i - \sum_{j \in Neighbors(i)} W_{ij} y_j|^2, \quad \sum_j W_{ij} = 1 \quad i = 1, \ldots, N \tag{4.14}$$

W specifies the local spatial relationships among samples in Y that have to be preserved in the embedding. Denote $G = (g_1, g_2, \ldots, g_N)$ as the LLE embedding result of Y. G are computed by minimizing the objective function:

$$\Phi(G) = \sum_i |g_i - \sum_j W_{ij} g_j|^2 = tr(G(I - W)^T (1 - W) G^T) \tag{4.15}$$

Note that $\Phi(G)$ is invariant to rotation and translation of G and its scale changes with the scale of G. In order to remove these degenerated conditions, additional constraints are imposed on G:

$$\sum_{i=1}^{N} g_i = 0, \text{ and } \sum_{i=1}^{N} g_i g_i^T = GG^T = I_q \tag{4.16}$$

Like LLE, the locally linear coordination (LLC) algorithm is also based on equations (4.14), (4.15) and (4.16). But in LLC, G is defined as $G = LU$ and the task is to compute L. By replacing G with LU in (4.15) and (4.16), L can computed by minimizing the objective function:

$$\Phi(L) = tr(LU(I - W)^T (1 - W) U^T L^T) = tr(LAL^T) \tag{4.17}$$

where $A = U(I - W)^T (1 - W) U^T$. Under the constraint that,

Locally Linear Coordination

Given data set $Y = \{y_1, \ldots, y_N\}$:

1. Learn the MPPCA model parameters Θ_{mppca} according to Section 3.2.2.
2. Compute $U = (u_1, \ldots, u_N)$ based on (4.12).
3. Compute W based on (4.14).
4. Compute matrix A and matrix B according to (4.17) and (4.18).
5. Solve the generalized eigenvalue problem for the matrix pair (A, B), and set the rows of L to be the 2nd to $(q+1)$-th smallest generalized eigenvectors of (A, B).

Table 4.2 The details of locally linear coordination algorithm

$$LUU^T L^T = LBL^T = I_q \tag{4.18}$$

with $B = UU^T$. The is a convex optimization problem, and the optimal solution can be obtained by solving the generalized eigenvalue problem for the matrix pair (A, B). Let l_k be the k-th smallest generalized eigenvector of (A, B), the solution of L is $L = (l_2, l_3, \ldots, l_{q+1})^T$. Table 4.2 summarizes this learning algorithm.

LLC algorithm like LLE is based on the locally spatial constraints defined in (4.14) and (4.15). In other words, the LLC result can only be as good as LLE. LLE similar to ISOMAP requires data in Y are densely sampling to ensure good embedding result, but this condition is seldom met in real applications.

4.3 Learning via EM

As is pointed out in Section 2.3.4, it is difficult to incorporate temporal constraints among samples into either ISOMAP or LLE. Therefore, neither Wang's or Teh's algo-

rithm can be directly extended to learn the global coordination model from temporal sequences. In this, we present an algorithm by Roweis et. al. [19] that learns global coordination model via an expectation-maximization (EM) approach. It is a general learning algorithm, and can be extended to learning from sequences of samples which will be explained in details in Chapter 5.

4.3.1 The Simplified Model

Before introducing the algorithm, we first demonstrate that the graphical model shown in Figure 4.1 can be simplified.

It is worth to note that variable z^s is not involved in the computation of $P(g|y,s)$ and $P(g|y,s)$ as are shown in (4.8) and (4.10). This is because when s is given, the mapping from z^s to g are deterministic and invertible as is defined in (4.4). According to this property and the definition $z^s \sim \mathcal{N}(0, I_q)$, $P(g|s)$ can be computed as:

$$P(g|s) = \int P(g|z^s, s)P(z^s|s)dz^s \sim \mathcal{N}(\kappa_s, A_s^{-T}A_s^{-1}) \tag{4.19}$$

At the same time, $P(y|g,s)$ can be computed using (4.10). Therefore, based on (4.19) and (4.10), the joint probability $P(y,g,s) = P(y|g,s)P(g|s)P(s)$ can be computed directly without involving z^s.

The property can also be represented graphically. Based on the original model shown in Figure 4.1, where both nodes z^s and g have links pointed from s. We can then merge node z^s into g, and Figure 4.2 shows the reduced model. By marginalizing out z^s from

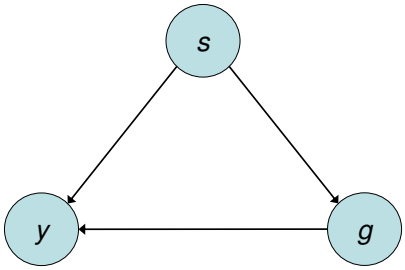

Figure 4.2 The reduced global coordination model. The node z^s shown in Figure 4.1 is merged into node g.

the graphical model, the number of parameters is reduced, which greatly facilitates the analysis of the following learning algorithm.

4.3.2 Learning

This EM approach algorithm is based on maximum likelihood estimation. Given observed data set $Y = \{y_1, \ldots, y_N\}$, the goal is to find $\Theta_{gc} = \{\Theta_{mppca}, \Theta_{align}\}$ that maximizes log-likelihood $\log P(Y)$. Following the graphical model shown in Figure 4.2, for observed sample y_i its hidden variables in the global coordination model are s_i and g_i. According to Jensen's inequality it can be shown that:

$$\mathcal{L} = \log P(Y) = \sum_{i=1}^{N} \log P(y_i) \geq \mathcal{F} = \sum_{i=1}^{N} \sum_{s_i=1}^{K} \int Q(s_i, g_i) \log \frac{P(y_i, s_i, g_i)}{Q(s_i, g_i)} dg_i \qquad (4.20)$$

Till this point, the derivation is identical to learning MPPCA in Section 3.2.2. This is not surprising since the global coordination model is an extension of MPPCA.

Now the question is how the distribution function $Q(s_i, g_i)$ shall be defined such that parameters Θ_{mppca} and parameters Θ_{align} can both be learned through Equation (4.20) via an EM-like algorithm. Ideally, given observed sample y_i, we expect that mapping to its global coordinates g_i is consistent thought all mixture components. That is,

$$P(g_i|y, s = a) \approx P(g_i|y, s = b), \quad a, b \in \{1, \ldots, K\} \tag{4.21}$$

If $Q(s_i, g_i)$ is set to be $P(g_i|s_i, y_i, \Theta_{gc})(s_i|y_i, \Theta_{gc})$ as in Section 3.2.2, this becomes the exact learning algorithm for MPPCA. Since there is no constraints to enforce (4.21), the result is $P(g_i|y, s)$ can be very different for each s. Therefore, the MPPCA learning algorithm fails to recover the intrinsic global coordinates of the manifold. According to (4.21), for each observed sample y_i it global coordinates g_i should be independent of s_i. Therefore, the distribution function $Q(s_i, g_i)$ can be factorized as:

$$Q(s_i, g_i) = Q(s_i)Q(g_i) \tag{4.22}$$

This factorization implicitly impose constraints of global coordinates in (4.20).

Since $Q(s_i, g_i)$ is no longer equal to $P(g_i|s_i, y_i, \Theta_{gc})(s_i|y_i, \Theta_{gc})$, the parametric distribution of $Q(g_i)$ and $Q(s_i)$ have to be defined before the EM-like algorithm can be started. Because $Q(s_i)$ is probability mass function, it is set as:

$$Q(s_i) = q_{is}, \quad \sum_s q_{is} = 1 \tag{4.23}$$

where q_{is} is a random variable, $0 \leq q_{is} \leq 1$. Probability density function $Q(g_i)$, on the other hand, is set to be an Gaussian:

$$Q(g_i) \sim \mathcal{N}(\bar{g}_i, \Sigma_i) \tag{4.24}$$

As is explained in Section 4.1.1, because the mapping from observed sample y to global coordinates g is expected to be one-to-one, $P(g_i|y_i)$ should be unimodel. This unimodal property is enforced in (4.24) by setting $Q(g_i)$ to be a Gaussian.

4.3.3 Algorithm Detail

With all the variables in (4.20) are well defined, this EM-based algorithm works iteratively as follows. In the E-step, parameters of the global coordination model Θ_{gc} are fixed, and parameters in $Q(g_i)$ and $Q(s_i)$ are updated subsequently to maximize $\mathcal{F}$ in (4.20). $Q(g_i)$ and $Q(s_i)$ then stay fixed in the M-step, while Θ_{gc} is updated to maximize $\mathcal{F}$. Note that Θ_{gc} is updated such that distribution $P(s_i, g_i|y_i)$ is close to $Q(g_i)$. In other words, Θ_{gc} is adjusted to satisfy the constraint of g defined in (4.22). The detailed algorithm is explained below. Starting with initial model parameters $\Theta_{gc}^{(0)}$, variables in $Q(s_i)$ are initialized as $q_{is} = P(s|y_i, \Theta_{gc}^{(0)})$ for all i and s.

The E-step first computes parameters in $Q(g_i)$ while all other parameters stay fixed. By taking the partial derivatives of (4.20) with respect to (g_i, Σ_i), it can be shown that $\mathcal{F}$ is maximized by setting (g_i, Σ_i) to be:

$$\Sigma_i = \left[\sum_s q_{is} V_s\right]^{-1}, \quad g_i = \Sigma_i \left[\sum_s q_{is} V_s g_{is}\right] \tag{4.25}$$

Where

$$\begin{aligned} V_s &= A_s^{-T}(I_d + \Lambda_s^T \Psi_s^{-1} \Lambda_s) A_s^{-1} \\ g_{is} &= E[P(g_i|y_i, s)] = A_s(I_d + \Lambda_s^T \Psi_s^{-1} \Lambda_s)^{-1} \Lambda^T \Psi^{-1} \tilde{y}_{is} + \kappa_s \\ \tilde{y}_{is} &= y_i - \mu_s \\ V_s g_{is} &= A_s^{-T} \Lambda^T \Psi^{-1} \tilde{y}_{is} + V_s \kappa_s \end{aligned}$$

After (g_i, Σ_i) is known, by taking partial derivatives of $\mathcal{F}$ with respect to $Q(s_i|y_i)$, the optimal q_{is} is:

$$q_{is} = \frac{e^{-\mathcal{E}_{is}}}{\sum_s e^{-\mathcal{E}_{is}}} \tag{4.26}$$

Where

$$\begin{aligned} \mathcal{E}_{is} &= -\int Q(g_i) \frac{P(y_i, g_i, s_i)}{Q(g_i)} dg_i \\ &= \frac{1}{2} g_{is}^T V_s g_{is} + \frac{1}{2} \tilde{y}_{is}^T \Psi^{-1} \tilde{y}_{is} - g_{is}^T A_s^{-T} \lambda_s^T \Psi_s^{-1} \tilde{y}_{is} + \frac{1}{2} tr(\Sigma_i V_s) \\ &\quad + \frac{1}{2} \log |\Psi_s| + \log |A_s| - \log \pi_s + \text{constant} \end{aligned}$$

There is a post-prossing procedure at the end of E-step. Since the objective function $\mathcal{F}$ is invariant to translation and scaling of g_i. To remove this degeneracy, set $\{g_i\}_i$ is translated and rescaled to become zero mean and unit variance in each direction.

The M-step updates model parameters Θ_{gc} according to $\{g_i, \Sigma_i, q_{is}\}_{i,s}$. The update equations for model parameters Θ_{gc} can be derived by taking the partial derivatives of $\mathcal{F}$ with respect to each parameter. Let $q_s = \sum_{i=1}^{N} q_i s$, parameters $\{\pi_s, \mu_s, \kappa_s\}$ are updated as follows:

$$\pi_s = \frac{q_s}{\sum_s q_s} \tag{4.27}$$

$$\mu_s = q_s^{-1} \sum_{i=1}^{N} q_{is} y_i \tag{4.28}$$

$$\kappa_s = q_s^{-1} \sum_{i=1}^{N} q_{is} g_i \tag{4.29}$$

To update the remaining parameters, we first compute $\tilde{y}_{is} = y_i - \mu_s$ and $\tilde{g}_{is} = g_i - \kappa_s$. We then compute correlation matrix $C_s = \sum_{i=1}^{N} q_{is} \tilde{y}_{is} \tilde{g}_{is}^T$ and covariance matrix $G_s = \sum_{i=1}^{N} q_{is}(\Sigma_i + \tilde{g}_{is}\tilde{g}_{is}^T)$. Based on these variables, $\{\Lambda_s, \Psi_s, A_s\}$ are updated accordingly,

$$\Lambda_s = C_s G_s^{-1} A_s \tag{4.30}$$

$$[\Psi_s]_i = q_s^{-1} \sum_{i=1}^{N} q_{is} \left\{ [\tilde{y}_{is} - \Lambda_s A_s^{-1} \tilde{g}_{is}]_i + [\Lambda_s A_s^{-1} \Sigma_i A_s^{-T} \Lambda_s^T]_i \right\} \tag{4.31}$$

$$A_s^{-1} = (I_d + \Lambda_s^T \Psi_s^{-1} \Lambda_s)^{-1} (A_s^T q_s + \Lambda_s \Psi_s^{-1} C_s) G_s^{-1} \tag{4.32}$$

This concludes one iteration of this EM-based algorithm. The iteration continues until convergence is reached.

4.3.4 Analysis

This EM-based algorithm incrementally maximizes the objective function to a local optimum. Experiments indicate that this algorithm has serious local optimal problem, and requires good initialization to ensure acceptable learning result. The parameters can be initialized using either Wang's algorithm in Sec 4.2.1 or Teh's algorithm in Sec 4.2.2. It is worth to note that unlike Wang's and Teh's algorithms which fix either the global coordinates or part of the model parameters during learning this EM-based algorithm updates both the model parameters and the global coordinates at each iteration throughout the learning process.

From (4.20), it can be shown that maximizing $\mathcal{F}$ is equivalent to minimizing the KL-divergence between $Q(s_i, g_i)$ and $P(s_i, g_i|y_i)$:

$$\mathcal{L} - \mathcal{F} = \sum_i \sum_s \int Q(s_i, g_i) \log \frac{Q(s_i, g_i)}{P(s_i, g_i|y_i)} dg_i = \sum_i KL(Q(s_i, g_i)||P(s_i, g_i|y_i)) \quad (4.33)$$

The KL divergence can be further factorized as:

$$KL(Q(s_i, g_i)||P(s_i, g_i|y_i)) = \sum_s q_{is} \int Q(g_i) \log \frac{Q(g_i)}{P(g_i|s_i, y_i)} dg_i + \sum_s q_{is} \log \frac{q_{is}}{P(s|y_i)} \quad (4.34)$$

and the first term on the high-hand side can be rewritten as:

$$\sum_s q_{is} \int Q(g_i) \log \frac{Q(g_i)}{P(g_i|s_i, y_i)} dg_i = \sum_s q_{is} KL(Q(g_i)||P(g_i|y_i, s_i)) \tag{4.35}$$

The term is maximized when $\{P(g_i|y_i, s_i = a) \sim P(g_i|y_i, s_i = b) \sim Q(g_i)|a, b \in 1, \ldots, K\}$. This is the spatial constraint imposed in the EM-based algorithm to algin the global coordinates.

In the original work by Roweis et. al. [19], the objective function to be maximized is defined as:

$$\mathcal{F} = \sum_i log(y_i) - \lambda \sum_i KL(Q(s_i, g_i)||P(s_i, g_i|y_i)) \tag{4.36}$$

which is to maximize the log likelihood $\log P(Y)$ under the global coordination constraints, and factor λ controls the effect of these constraints. In our derivation of this EM-based algorithm, λ is set to one. This alignment constraint is also adopted in other learning algorithm for global coordination model. Verbeek et. al. [22] propose a two stage approach similar to Teh's algorithm, but they use (4.35) to learn the alignment parameters.

The global coordination constraint in(4.35) is difficult to be satisfied. As can be shown, $P(g_i|y_i, s_i = a) = P(g_i|y_i, s_i = b)$ happens when the two PPCA models, a and b, lie on the same subspace. To ensure $P(g_i|y_i, s_i = a) \sim P(g_i|y_i, s_i = b)$, the principal angles between the two subspaces should be small. As a result, when fitting a nonlinear manifold, a large number of mixture components will be needed to make sure there is slow

principal angle variation across adjacent PPCA models. This problem is also pointed out in [20].

4.4 Summary

In the chapter, we introduce the global coordination (GC) model and survey three different approaches to learning the GC model from i.i.d. data set. Since learning the GC model from the i.i.d. data set is a ill-posed problem, neither one of these algorithms provides satisfactory results

In the next chapter, we present our algorithm to learn the GC model from sequences. Experimental results demonstrate that our algorithm achieve superior learning results.

CHAPTER 5

Learning Global Coordination Model Through Temporal Sequences

In the chapter, we introduce the theme of this work: learning global coordination model trough temporal sequences. As is pointed out in the previous chapter, learning global coordination model trough independent, identically distributed (i.i.d.) samples is ill-posed and has serious local minimal problem. Here we reformulate the learning problem by assuming that the training data are temporal sequences and each sample is highly correlated with theirs temporal neighbors. This assumption, as matter of fact, fits to many practical applications, since in the real world the perception is always continuous and there exists strong temporal dependency among the successive percepts.

In order to take the temporal information into account, we extend the global coordination model into a state-space model in which the continuous state variables correspond to the global coordinates. This extension expands the learning task. Now we have to learn a complete state-space model with the global coordination model being the state-output mapping function.

We start with a simple state-space model in which the transition of state variables is based on linear dynamics. Even with this simplification, exact inference in our model is intractable. Here we propose an approximate inference algorithm, based on which we derive an efficient variational algorithm to learn the model parameters. We then experiment our algorithm using synthetic data set, and demonstrate that our model achieves superior results than the global coordination model trained with i.i.d. samples. We then illustrate that by a small modification our model become a nonlinear state-space model which has nonlinear state transition dynamics and nonlinear state-to-output mapping, and it can learned under the same variational framework. This chapter concludes the theoretical work of this dissertation. Experiments of our model on real world applications will be presented in the next chapter.

5.1 Dynamic Global Coordination Model

In the previous chapters, the data sets used for manifold learning are assumed to be a collection of i.i.d. samples. In this chapter, we formulate a different manifold learning problem in which the data set are temporal sequences.

Let $Y = \{y_1, y_2, \ldots, y_T\}$ be a high-dimensional observation sequence of length T. Y is generated according to a underlying state variable sequence $G = \{g_1, g_2, \ldots, g_T\}$ such that sample y_t is mapped from state variables g_t using function $f(\cdot)$:

$$y_t = f(g_t) + u_t \tag{5.1}$$

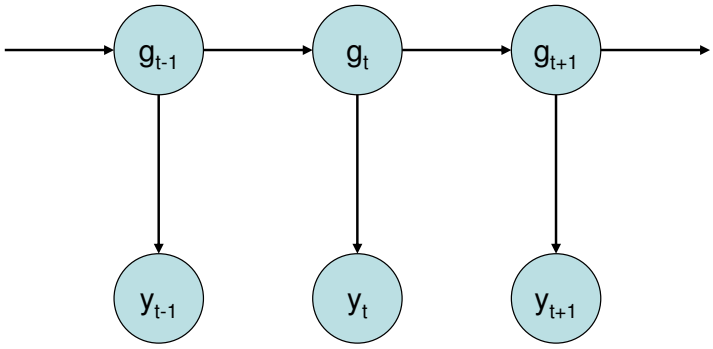

Figure 5.1 The dynamic model that generates the observation sequence.

where u_t is a Gaussian noise. In addition, we assume sequence G is generated according to a Markovian dynamic model, $h(\cdot)$:

$$g_{t+1} = h(g_t) + v_t \tag{5.2}$$

with v_t denoting another Gaussian noise. According to Equation (5.1) and Equation (5.2), the relationship between sequence Y and sequence G can be described by a state-space model shown in Figure 5.1.

Learning a state-space model is to estimate the parameters of the mapping function $f(\cdot)$, the state transition function $h(\cdot)$, and the two noise models. For our learning problem, the mapping from g_t to y_t is provided by the global coordination model described in chapter 4, and we set the state transition in G to be based on a linear dynamic model with Gaussian noise:

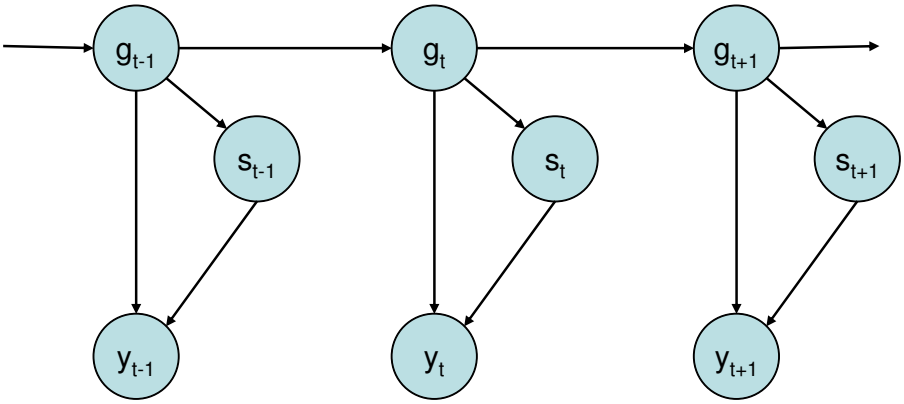

Figure 5.2 Our dynamic global coordination model.

$$g_{t+1} = Bg_t + v_t \tag{5.3}$$

where $v_t \sim \mathcal{N}(0, R)$. Therefore,learning this state-space model is equivalent to learning the global coordination model and the linear dynamic model simultaneously. If we represent the mapping from g_t to y_t using the global coordination model, the resulting graphical model is shown in Figure 5.2.

Note that the link connecting g_t and s_t is pointing from g_t to s_t, because in this dynamic process sequence $G = \{g_t\}_t$ is solely generated by the Markovian dynamic model. This dynamic global coordination generates an observation sequence, $Y = \{y_t\}_{t=1}^T$, by taking the following steps. Firstly, sequence $G = \{g_t\}_{t=1}^T$ is produced using the given Markovian dynamic model. Then, for each g_t, a latent variable s_t is selected according to the distribution function, $P(s_t|g_t)$. Based on variable s_t, the linear function that maps

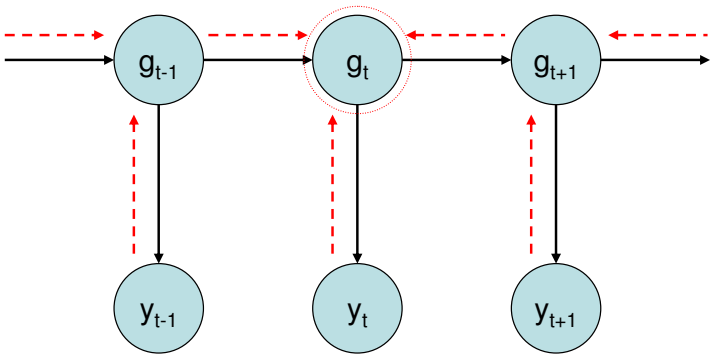

Figure 5.3 The inference process of our dynamic global coordination model.

g_t to y_t is chosen, and observation y_t is produced thereafter. For most real applications, the focus is on the relationship between G and Y. As a result, latent variables $\{s_t\}$ are usually marginalized out during the inferences.

It is worth to note that there is an apparent difference between the global coordination model and our dynamic global coordination model, while inferencing g_t from observation Y. The global coordination model assumes samples in Y are i.i.d., so g_t is solely determined by y_t. On the contrary, in our dynamic global coordination model, Y represents a temporal correlated sequences and g_t is determined by the whole sequence, Y. This is illustrated in Figure 5.3 with the red dot lines indicating the pathes to propagate the information of all elements in Y to g_t.

5.2 Approximate Algorithm

The exact inference of the graphical model shown in Figure 5.3 is intractable due to its innate model complexity. In this chapter we propose an approximate variational inference and learning algorithm for it. Our proposed algorithm is based on two strong assumptions, which makes our algorithm less general and will work only in certain conditions.

Our first assumption is that the distribution the distribution $P(g_t|t_t)$ can be approximated as a Gaussian. As will be seen in the following sections, this property is a key component to make our algorithm work. According to equation (4.6), if $P(s_t = s|y_t)$ is nonzero for a particular s, $P(g_t|y_t)$ will be a Gaussian. Note that because the global coordination model is an extension of the mixtures of probabilistic principal analyzers $P(s_t|y_t)$ is solely determined by the MPPCA module. Therefore, if the training data set can be fitted by a MPPCA such that the distribution $P(s_t = s|y_t)$ is close to be a delta function, our unimodal assumption is not violated. We examine the synthetic and real world data used in our experiments, and this property is satisfied in all the data set we use.

Our second assumption imposes an even stronger constraint. In the following section, we assume probability $P(g_t)$ is negligible. Therefore, $P(s_t|g_t)$ can be approximated by the joint $P(s_t, g_t)$. Neglecting the effect of $P(g_t)$ is susceptible to failure and has to be applied carefully. However, in our variational learning algorithm neglecting the effect of $P(g_t)$ can be interpreted as including an additional regularization term. That is, instead

of optimizing $\log P(y_{1:T})$, we are optimizing $\log P(y_{1:T}) + \sum_{i=1}^{t} \log P(g_i)$. By imposing the additional constraint through the regularization term, $\sum_{i=1}^{t} logP(g_i)$, the inference becomes management. In our algorithm, we ensure that the approximate approach is consistent during learning and inference. That is, this approximate approach is applied to both the inference and learning processes.

Given these two preconditions, we can now derive our inference and learning algorithm.

5.3 Inference

We now provide the inference algorithm for our dynamic global coordination model. Given the observation sequence $\{y_1, \ldots, y_t\}$ denoted as $y_{1:t}$, we want to estimate the posterior probability distribution $P(g_t|y_{1:t})$. According to the Bayes filter, $P(g_t|y_{1:t})$ can be factorized as:

$$\begin{aligned} P(g_t|y_{1:t}) &\propto P(y_t|g_t)P(g_t|y_{1:t-1}) \\ &= P(y_t|g_t) \int P(g_t|g_{t-1})P(g_{t-1}|y_{1:t-1})dg_{t-1} \\ &= \sum_{s_t} P(y_t|g_t, s_t)P(s_t|g_t) \int P(g_t|g_{t-1})P(g_{t-1}|y_{1:t-1})dg_{t-1} \end{aligned} \tag{5.4}$$

As is shown in Section 4.1, $P(y_t|g_t)$ is a mixture of Gaussians. Hence, the number of mixtures in $P(g_t|y_{1:t})$ will exponentially grow with t, which makes the exact inference of $P(g_t|y_{1:t})$ intractable and an approximate method needed.

In Section 4.1.1, we have pointed out that even though $P(y_t|g_t)$ takes the form of a mixture model its actual distribution is closed to unimodel. According to this property, we can approximate $P(y_t|g_t)$ as a unimodal Gaussian distribution, and develop an efficient approximate inference algorithm.

5.3.1 Approximate Unimodal Distribution

We first present our method to approximate $P(g_t|y_t)$ as a single Gaussian. According to (4.6), $P(g_t|y_t) = \sum_{s_t} P(g_t|y_t, s_t)P(s_t|y_t)$ in which $P(g_t|y_t, s_t) \sim \mathcal{N}(\mu_t^s, \Sigma_t^s)$. Let (μ_t, Σ_t) denote the mean and the covariance matrix of the Gaussian that we use to approximate $P(g_t|y_t)$. (μ_t, Σ_t) can be estimated according to the first order Generalized Pseudo Bayesian (GPB) algorithm [55][56], which is the best single Gaussian approximation in the KL sense:

$$(\mu_t, \Sigma_t) = \arg\min_{\mu,\Sigma} \sum_s P(s_t|y_t) KL(\mathcal{N}(\mu_t^s, \Sigma_t^s)||\mathcal{N}(\mu, \Sigma)). \tag{5.5}$$

where $KL(\mathcal{N}(\mu_t^s, \Sigma_t^s)||\mathcal{N}(\mu, \Sigma))$ denotes the KL-divergence between the two distributions.

Since the both distributions in $KL(\mathcal{N}(\mu_t^s, \Sigma_t^s)||\mathcal{N}(\mu, \Sigma))$ are Gaussians, it can be shown that:

$$\begin{aligned} &KL(\mathcal{N}(\mu_t^s, \Sigma_t^s)||\mathcal{N}(\mu, \Sigma)) = \int \mathcal{N}(g_t; \mu_t^s, \Sigma_t^s) \log \frac{\mathcal{N}(g_t; \mu_t^s, \Sigma_t^s)}{\mathcal{N}(g_t; \mu, \Sigma)} dg_t \\ &= \frac{1}{2}\left(|\Sigma| - |\Sigma_t^s| + trace(\Sigma^{-1}\Sigma_t^s) + (\mu_t^s - \mu)^T\Sigma^{-1}(\mu_t^s - \mu) - trace(\Sigma_t^{s-1}\Sigma_t^s)\right) \end{aligned} \tag{5.6}$$

Applying (5.6) to (5.5) and taking derivatives with respect to μ and Σ, we can obtain the analytic optimal solution of (μ_t, Σ_t):

$$\mu_t = \sum_s P(s_t|y_t)\mu_t^s \tag{5.7}$$

$$\Sigma_t = \sum_s P(s_t|y_t)\left(\Sigma_t^s + (\mu_t - \mu_t^s)(\mu_t - \mu_t^s)^T\right) \tag{5.8}$$

Based on this result, we then derive the approximate inference algorithm for filtering and smoothing in our dynamic global coordination model.

5.3.2 Filtering

Filtering is to estimate g_t based on observation $y_{1:t}$. As is shown in (5.4), $P(g_t|y_{1:t})$ is proportional to $P(y_t|g_t)P(g_t|y_{1:t-1})$. By applying the Bayes rule to $P(y_t|g_t)$, we get:

$$P(y_t|g_t) = \frac{P(g_t|y_t)P(y_t)}{P(g_t)} \tag{5.9}$$

Since y_t is observed, $P(y_t)$ is an constant that can be ignored. In our work, we assume the distribution of $P(g_t)$ is relatively flat and neglect the effect of $P(g_t)$ in (5.9). That is, we approximately $P(y_t|g_t)$ with $P(g_t|y_t)$. We will revisit this flat distribution assumption while introducing our learning algorithm, and provide detailed analysis there. Based on the results in (5.7) and (5.8), $P(g_t|y_t)$ is approximated as a single Gaussian. Therefore, we obtain the approximation,

$$P(y_t|g_t) \propto \mathcal{N}(\mu_t, \Sigma_t) \tag{5.10}$$

According to (5.3), $P(g_t|g_{t-1}) \sim \mathcal{N}(Bg_{t-1}, R)$. Since $P(g_t|g_{t-1})$ and $P(y_t|g_t)$ are both Gaussians, the posterior distribution $P(g_t|y_{1:t})$ will also be a Gaussian. Therefore, the filtering process in our model is very similar toKalman filter [57]. Let $P(g_t|y_{1:t}) \sim \mathcal{N}(\mu_t^t, \Sigma_t^t)$. (μ_t^t, Σ_t^t) can be computed iteratively. Given $(\mu_{t-1}^{t-1}, \Sigma_{t-1}^{t-1})$, it can be shown that,

$$\begin{aligned} P(g_t|y_{1:t-1}) &= \int P(g_t|g_{t-1})P(g_{t-1}|y_{1:t-1})dg_{t-1} \sim \mathcal{N}(\mu_t^{t-1}, \Sigma_t^{t-1}) \\ g_t^{t-1} &= Bg_{t-1}^{t-1} \qquad (5.11) \\ \Sigma_t^{t-1} &= B\Sigma_{t-1}^{t-1}B^T + R \qquad (5.12) \end{aligned}$$

Based on the result of $P(g_t|y_{1:t-1})$ and (5.10), we obtain:

$$\begin{aligned} P(g_t|y_{1:t-1}) &\sim \mathcal{N}(\mu_t^t, \Sigma_t^t) \propto P(y_t|g_t)P(g_t|y_{1:t-1}) \\ \Sigma_t^t &= \left((\Sigma_t^{t-1})^{-1} + \Sigma_t^{-1}\right)^{-1} \qquad (5.13) \\ g_t^t &= \Sigma_t^t\left((\Sigma_t^{t-1})^{-1}g_t^{t-1} + \Sigma_t^{-1}\mu_t\right) \qquad (5.14) \end{aligned}$$

Note that unlike Kalman filter which uses a fixed Gaussian for the measurement function $P(y_t|g_t)$, in our dynamic global coordination model (μ_t, Σ_t) are adaptively updated according to y_t as is in shown in (5.7) and (5.8).

5.3.3 Smoothing

In smoothing, we are computing $P(g_t|y_{1:T}) \sim \mathcal{N}(\mu_t^T, \Sigma_t^T)$ with $T \geq t$. The derivation of smoothing is based on the property of Gaussian distribution. Given $P(g_t, g_{t-1}|y_{1:T})$ is a Gaussian, it can be proved that:

$$\mu_{t-1}^T = E[g_{t-1}|y_{1:T}] = E[g_{t-1}|y_{1:T}, g_t = \mu_t^T] \tag{5.15}$$

Hence, μ_{t-1}^T can be computed from the parametric distribution $P(g_{t-1}|g_t, y_{1:T})$.

We first decompose $P(g_t, g_{t-1}|y_{1:T})$ as:

$$\begin{aligned} P(g_t, g_{t-1}|y_{1:T}) &= \frac{P(y_{t:T}|g_t, g_{t-1}, y_{t-1})P(g_t|g_{t-1}, y_{t-1})P(g_{t-1}|y_{1:t-1})}{P(y_{1:T})} \\ &= \frac{P(y_{t:T}|g_t)P(g_t|g_{t-1})P(g_{t-1}|y_{1:t-1})}{P(y_{1:T})} \\ &= k_1(g_t, y_{1:T})P(g_t|g_{t-1})P(g_{t-1}|y_{1:t-1}) \end{aligned} \tag{5.16}$$

Following the result in (5.16), $P(g_{t-1}|g_t, y_{1:t})$ can be factorized as:

$$\begin{aligned} P(g_{t-1}|g_t, y_{1:t}) &= \frac{P(g_t, g_{t-1}|y_{1:T})}{P(g_t|y_{1:T})} \\ &= k_2(g_t, y_{1:T})P(g_t|g_{t-1})P(g_{t-1}|y_{1:t-1}) \\ &= k_2(g_t, y_{1:T})\mathcal{N}(g_{t-1}; \tilde{\Sigma}_{t-1}([\Sigma_{t-1}^{t-1}]^{-1}\mu_{t-1}^{t-1} + B^T R^{-1} y_t), \tilde{\Sigma}_{t-1}) \end{aligned} \tag{5.17}$$

with $\tilde{\Sigma}_{t-1} = ([\Sigma_{t-1}^{t-1}]^{-1} + B^T R^{-1} B)^{-1}$. According to (5.17) and (5.15), by setting $y_t = \mu_t^T$ we get,

$$g_{t-1}^T = ([\Sigma_{t-1}^{t-1}]^{-1} + B^T R^{-1} B)^{-1}([\Sigma_{t-1}^{t-1}]^{-1}\mu_{t-1}^{t-1} + B^T R \mu_t^T) \tag{5.18}$$

In addition, it can be proved that the following properties holds,

$$([\Sigma_{t-1}^{t-1}]^{-1} + B^T R^{-1} B)^{-1} = [\Sigma_{t-1}^{t-1}] - [\Sigma_{t-1}^{t-1}]B^T(B[\Sigma_{t-1}^{t-1}]B^T + R)^{-1}B[\Sigma_{t-1}^{t-1}] \tag{5.19}$$

$$([\Sigma_{t-1}^{t-1}]^{-1} + B^T R^{-1} B)^{-1} B^T R^{-1} = [\Sigma_{t-1}^{t-1}]B^T(B[\Sigma_{t-1}^{t-1}]B^T + R)^{-1} \tag{5.20}$$

Applying (5.19), (5.20), (5.11) and (5.12) to (5.18), it follows that:

$$\mu_{t-1}^T = \mu_{t-1}^{t-1} + J_{t-1}(\mu_t^T - \mu_t^{t-1}) \tag{5.21}$$

$$J_{t-1} = \Sigma_{t-1}^{t-1} B^T [\Sigma_t^{t-1}]^{-1} \tag{5.22}$$

Which are the smoothing equations for the mean.

From (5.21), we get:

$$(y_t - \mu_{t-1}^T) + J_{t-1}\mu_t^T = (y_t - \mu_{t-1}^{t-1}) + J_{t-1}\mu_t^{t-1} \tag{5.23}$$

Multiplying both sides with their respective transposes from the right and after some manipulation of equations:

$$\Sigma_{t-1}^{t} + J_{t-1}\mu_t^T(\mu_t^T)^T J_{t-1}^T = \Sigma_{t-1}^{t-1} + J_{t-1}B\mu_{t-1}^{t-1}(\mu_{t-1}^{t-1})^T B^T J_{t-1}^T \tag{5.24}$$

It can be shown that:

$$\mu_t^T(\mu_t^T)^T = BE[y_{t-1}y_{t-1}^T]B^T + R - \Sigma_{t-1}^T \tag{5.25}$$

$$\mu_{t-1}^{t-1}(\mu_{t-1}^{t-1})^t = E[y_{t-1}y_{t-1}^T] - \Sigma_{t-1}^{t-1} \tag{5.26}$$

Putting these together with (5.24), we obtain the smoothing equation for the covariance matrix:

$$\Sigma_t^T = \Sigma_t^t + J_t\left(\Sigma_{t+1}^T - \Sigma_{t+1}^t\right)J_t^T \tag{5.27}$$

It is interesting to know that after the lengthy derivation, our smoothing recursive equations are identical to those in Kalman filter. The is attributed to the fact that these smoothing equations depend only on the filtering results, $\{\mu_t^t, \Sigma_t^t\}_{t=1}^T$, and dynamic model, $\{B, R\}$ in (5.3). Our dynamic model computes $\{\mu_t^t, \Sigma_t^t\}_{t=1}^T$ differently, but has the same linear dynamics as a Kalman filter. Hence, even though our dynamic model has a more complex state-output mapping, the smoothing equations are the same.

In implementation, the smoothing procedure starts from $t = T$ with (μ_T^T, Σ_T^T) being filtering result of $P(g_T|y_{1:T})$. It then iteratively decrements the value of t and compute (μ_t^T, Σ_t^T) using (5.21), (5.22) and (5.27). During smoothing, we also compute the lag-one covariance smoother, $\Sigma_{t,t-1}^T = E\left[\tilde{g}_t \tilde{g}_{t-1}^T | y_{1:T}\right]$ with $\tilde{g}_t = g_t - \mu_t^T$, which can be computed by the following recursive equation:

$$\Sigma_{t,t-1}^T = \Sigma_t J_{t-1}^T + J_t \left(\Sigma_{t+1,t}^T - C\Sigma_t^t\right) J_{t-1}^T \tag{5.28}$$

It should be emphasized that although our filtering and smoothing procedures are similar to Kalman filter, we have a more a more generalized dynamic model. In Kalman filter, the mapping from state variables to output is linear and static. Hence, Kalman filter is a model for dynamic inferences on a linear manifold. On the other hand, in our dynamic model, we use the global coordination model to provide the mapping from state to output. Therefore, our model performs dynamic inference on a nonlinear manifold.

5.4 Variational Learning

We take a variational approach to learn our dynamic model. The model parameters to be estimated are $\Theta_{dgc} = \{\Theta_{gc}, \Theta_{dm}\}$ with $\Theta_{gc} = \{\pi_s, \Lambda_s, \mu_s, \Psi_s, A_s, \kappa_s\}_s$ being the parameters in the global coordination model and $\Theta_{dm} = \{B, R\}$ being the parameters in the dynamic models.

To simplify the analysis, we assume the observation is a single sequence. However, our algorithm presented in the following can be easily generalized for multiple sequences.

Given an observation sequence $y_{1:T}$, we want to find Θ_{dgc} such that the likelihood $P(y_{1:T})$ is maximized. Using Jensen's inequality,

$$\log P(y_{1:T}) \geq \mathcal{F} = \sum_{s_{1:T}} \int Q(g_{1:T}, s_{1:T}) \log \left(\frac{P(y_{1:T}, g_{1:T}, s_{1:T})}{Q(g_{1:T}, s_{1:T})} \right) dg_{1:T} \tag{5.29}$$

Following the variational approach in Chapter 4, we factorize $Q(g_{1:T}, s_{1:T})$ into two distribution functions:

$$Q(g_{1:T}, s_{1:T}) = Q(s_{1:T})Q(g_{1:T}) \tag{5.30}$$

In our work, we factorize $Q(s_{1:T})$ as:

$$Q(s_{1:T}) = \prod_t Q(s_t) = \prod_t q_{ts} \tag{5.31}$$

where $0 \leq q_{ts} \leq 1$ and $\sum_s q_{ts} = 1$. As for $Q(g_{1:T})$, we define $Q(g_t)$ be a Gaussian:

$$Q(g_t) = \int Q(g_{1:T}) dg_1 \ldots dg_{t-1} dg_{t+1} \ldots dg_T \sim \mathcal{N}(\mu_t^Q, \Sigma_t^Q) \tag{5.32}$$

Out variation algorithm works as follows. Starting with the initial value $\Theta_{dgc}^{(0)}$, in the E-step we interleave the update of q_{ts} and (μ_t^Q, Σ_t^Q) to maximize $\mathcal{F}$. In the M-step, according to q_{ts} and (μ_t^Q, Σ_t^Q) we then compute the optimal Θ_{dgc}. This iterative procedure continues until it reaches convergence.

5.4.1 E-step

We first show the update of $\{q_{ts}\}_t$ with given $\{Q(g_t)\}_t$ and Θ_{dgc}. According to Figure 5.2, $P(y_{1:T}, g_{1:T}, s_{1:T})$ can be factorized as:

$$P(y_{1:T}, g_{1:T}, s_{1:T}) = P(g_1) \prod_{t=2}^{T} P(g_t|g_{t-1}) \prod_{t=1}^{T} P(s_t|g_t) \prod_{t=1}^{T} P(y_t|s_t, g_t) \tag{5.33}$$

Like our inference algorithm, for each $P(s_t|g_t)$ we neglect the effect of $P(g_t)$ and approximate it using the joint probability $P(g_t, s_t)$.

Plugging this result into (5.29), it follows that,

$$\begin{aligned}
\mathcal{F} &= \sum_{s_{1:T}} Q(s_{1:T}) \int Q(g_{1:T}) \log P(s_{1:T}, g_{1:T}, y_{1:T}) dg_{1:T} \\
&\quad - \sum_{s_{1:T}} Q(s_{1:T}) \log Q(s_{1:T}) - \int Q(g_{1:T}) \log Q(g_{1:T}) dg_{1:T} \\
&= \sum_{s_{1:T}} Q(s_{1:T}) \left(\sum_{t=1}^{T} \int Q(g_t) \log P(y_t, g_t, s_t) dg_t + \sum_{t=2}^{T} \int Q(g_t, g_{t-1}) \log P(g_t|g_{t-1}) \right. \\
&\quad \left. + \int Q(g_1) \log P(g_1) dg_1 \right) - \sum_{s_{1:T}} Q(s_{1:T}) \log Q(s_{1:T}) - \int Q(g_{1:T}) \log Q(g_{1:T}) dg_{1:T} \\
&= \sum_{t=1}^{T} \sum_{s=1}^{S} q_{ts} \int Q(g_t) \log P(y_t, g_t, s_t) dg_t + \sum_{t=2}^{T} \int Q(g_t, g_{t-1}) \log P(g_t|g_{t-1}) dg_t dg_{t-1} \\
&\quad + \int Q(g_1) \log P(g_1) dg_1 - \sum_{t=1}^{T} \sum_{s=1}^{S} q_{ts} \log q_{ts} - \int Q(g_{1:T}) \log Q(g_{1:T}) dg_{1:T}
\end{aligned} \tag{5.34}$$

By taking the derivative of $\mathcal{F}$ with respect to q_{ts}, the optimal q_{ts} that maximizes $\mathcal{F}$ is:

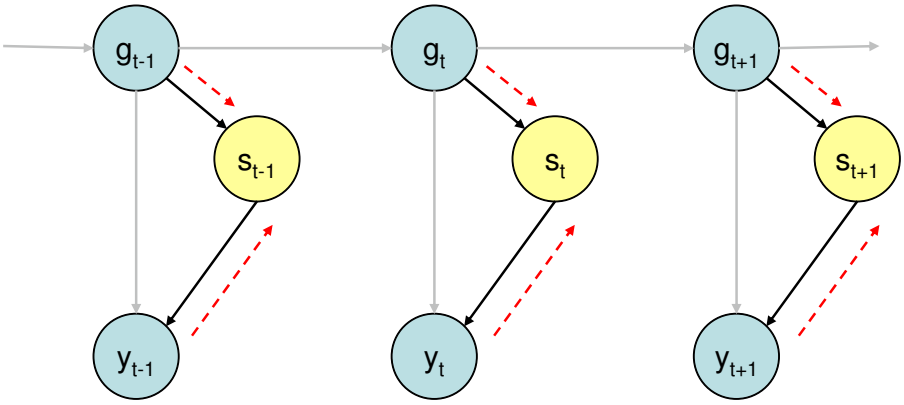

Figure 5.4 The inference process to compute q_{ts} during the E-step.

$$\begin{aligned} q_{ts} &= \frac{\exp(-\mathcal{E}_{ts})}{\sum_s \exp(-\mathcal{E}_{ts})} \\ \mathcal{E}_{ts} &= \int Q(g_t) \log P(y_t, g_t, s_t) dg_t \end{aligned} \tag{5.35}$$

Note that this result in (5.35) is identical to the update of q_{is} in the global coordination model shown in (4.26). This is no surprise since when $Q(g_t)$ is known, q_{ts} is solely determined by $Q(g_t)$ and y_t. The temporal dependency between $\{g_t\}_t$ has no effect here and can be neglected as is shown in Figure 5.4. However, we would like to emphasize that $Q(g_t)$ in our dynamic model is determined by the whole observation sequence $y_{1:T}$, unlike the $Q(g_t)$ in the global coordination model whose value depends on the single observation y_t.

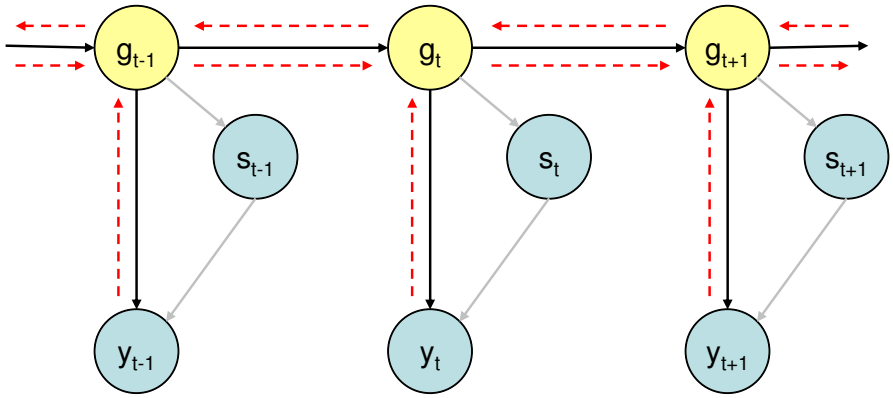

Figure 5.5 The inference process to compute (μ_t^Q, Σ_t^Q).

Once the set $\{g_{ts}\}_{t,s}$ is known, we can fix it and update $\{(\mu_t^Q, \Sigma_t^Q)\}_t$. It is difficult to directly estimate $\{(\mu_t^Q, \Sigma_t^Q)\}_t$ from (5.34), and we adopt an alterative approximate method. We start with a special case by assuming both $y_{1:T}$ and $s_{1:T}$ are observed. Since s_t is now deterministic,

$$y_t = \Lambda_{s_t} g_t + \mu_{s_t} + \epsilon_t \tag{5.36}$$

where $\epsilon_t \sim \mathcal{N}(0, \Psi_{s_t})$. This is an extended Kalman filter in which the linear state-output dynamically changes, and we can compute $P(g_t|y_{1:T}, s_{1:T})$ using the standard Kalman filtering and smoothing algorithm.

Back to the original problem. Since $q_{ts} \simeq P(s_t = s|y_{1:T})$, q_{ts} represents the weight of mapping g_t to y_t using (Λ_s, μ_s). Based on the this result, we define $\hat{P}(g_t|y_t)$ as:

$$\hat{P}(g_t|y_t) = \sum_s q_{ts} P(g_t|y_t, s) \tag{5.37}$$

and we approximate it as a Gaussian, $\hat{P}(g_t|y_t) \sim \mathcal{N}(\mu_t, \Sigma_t)$ using our algorithm in Section 5.3.1, it then follows that,

$$\mu_t = \sum_s q_{ts} \mu_t^s \tag{5.38}$$

$$\Sigma_t = \sum_s q_{ts} \left(\Sigma_t^s + (\mu_t - \mu_t^s)(\mu_t - \mu_t^s)^T \right) \tag{5.39}$$

After $\{(\mu_t, \Sigma_t)\}_t$ are known, we then apply the filtering and smoothing described in Section 5.3.2 and Section 5.3.3 to compute $\{(\mu_t^T, \Sigma_t^T)\}_t$. We then set $\mu_t^Q = \mu_t^T$ and $\Sigma_t^Q = \Sigma_t^T$. This inference procedure is depicted in Figure 5.5. Note that the links from g_t and y_t disappear since the distribution of s_t has been determined by $\{q_{ts}\}_s$. The filtering/smoothing approach to compute $\{(\mu_t^Q, \Sigma_t^Q)\}_t$ in our algorithm shares the resemblance to the variational learning algorithm for switching state-space models by Ghahramani et. al. [58]. However, since our dynamic graphical model is disparate from theirs, the detailed algorithm is completely different.

In actual implementation, we start the E-step by setting $q_{ts} = P(s|y_t)$ for all t and s, and use the $\{q_{ts}\}_{t,s}$ to update $\{(\mu_t^Q, \Sigma_t^Q)\}_t$. Then, based on the new $\{(\mu_t^Q, \Sigma_t^Q)\}_t$, the set $\{q_{ts}\}_{t,s}$ are updated. This iterative procedure continues until convergence.

5.4.2 M-Step

In the M-step, the model parameters $\Theta_{dgc} = \{\Theta_{gc}, \Theta_{dm}\}$ are updated based with the newly obtained $\{q_{ts}\}_{t,s}$ and $\{(\mu_t^Q, \Sigma_t^Q)\}_t$.

For the dynamic model parameters $\Theta_{dm} = \{B, R\}$, by taking the derivative of $\mathcal{F}$ with respect to B and R it can be shown that:

$$B_{new} = \left[\sum_{t=2}^{T} D_{t,t-1}^T\right]\left[\sum_{t=2}^{T} D_{t-1}^T\right]^{-1} \tag{5.40}$$

$$R_{new} = \frac{1}{T-1}\sum_{t=2}^{T}(D_t^T - B_{new}D_{t,t-1}) \tag{5.41}$$

where $D_{t,t-1} = \Sigma_{t,t-1}^T + (\mu_t^T)(\mu_{t-1}^T)^T$ and $D_t^T = \Sigma_t^T + (\mu_t^T)(\mu_t^T)^T$.

As for Θ_{gc}, since $\{q_{ts}\}_{t,s}$ and $\{(\mu_t^Q, \Sigma_t^Q)\}_t$ are known, Θ_{gc} can be computed according to the variational learning of the global coordination model shown in Section 4.3.3. Let $q_s = \sum_{t=1}^{N} q_t s$, parameters $\{\pi_s, \mu_s, \kappa_s\}$ are updated as follows:

$$\pi_s = \frac{q_s}{\sum_s q_s} \tag{5.42}$$

$$\mu_s = q_s^{-1}\sum_{t=1}^{T} q_{ts} y_t \tag{5.43}$$

$$\kappa_s = q_s^{-1}\sum_{t=1}^{T} q_{ts}\mu_t^Q \tag{5.44}$$

Variational Learning of the Dynamic Global Coordination Model

Given data set $Y = \{y_1, \ldots, y_N\}$ and the initial estimate of $\{q_{ts}\}_{t,s}$.

The E-step:

1. Compute $\{(\mu_t, \Sigma_t)\}_t$ according to (5.38) and (5.39).
2. Compute $\{(\mu_t^T, \Sigma_t^T)\}_t$ by applying the smoothing algorithm described in Section 5.3.3.
3. Update $\{q_{ts}\}_{t,s}$ using (5.35) and repeat the M-step until converges.

The M-step:

1. Update the parameters for the dynamic dynamic model according to (5.40) and (5.41).
2. Update the parameters for the global coordination model according to (5.42), (5.43), (5.44), (5.45), (5.46) and (5.47).

Table 5.1 Our variational based learning algorithm for the dynamic global coordination model.

For the remaining parameters, denote $\tilde{y}_{is} = y_i - \mu_s$ and $\tilde{g}_{ts} = \mu_t^Q - \kappa_s$. We then define $C_s = \sum_{t=1}^T q_{ts}\tilde{y}_{ts}\tilde{g}_{ts}^T$ and $G_s = \sum_{t=1}^T q_{ts}(\Sigma_t^Q + \tilde{g}_{ts}\tilde{g}_{ts}^T)$. Based on these, $\{\Lambda_s, \Psi_s, A_s\}$ are updated accordingly:

$$\Lambda_s = C_s G_s^{-1} A_s \tag{5.45}$$

$$[\Psi_s]_i = q_s^{-1} \sum_{i=1}^N q_{is} \left\{ [\tilde{y}_{is} - \Lambda_s A_s^{-1} \tilde{g}_{is}]_i + [\Lambda_s A_s^{-1} \Sigma_i A_s^{-T} \Lambda_s^T]_i \right\} \tag{5.46}$$

$$A_s^{-1} = (I_d + \Lambda_s^T \Psi_s^{-1} \Lambda_s)^{-1} (A_s^T q_s + \Lambda_s \Psi_s^{-1} C_s) G_s^{-1} \tag{5.47}$$

Our complete learning algorithm is summarized in Table 5.1

We want to emphasize that our learning result will be different from the global coordination model since we estimates $Q(g_t)$ differently. While the global coordination model

assumes observation samples are i.i.d, and $Q(g_t)$ is only related to y_t, our estimation of $Q(g_t)$ is based on the whole observation sequence $y_{1:T}$. That is, our model is developed within a dynamic context in which temporal correlation is taken into consideration.

Our dynamic model expand the Kalman filter framework for dynamic inference on a nonlinear manifold, but by complicating the model the exact inference in our model is infeasible. Our algorithm presented here provides an alternative solution to both the inference and learning problems.

In our approach, we use the joint probability $P(s_t, g_t)$ instead of $P(s_t|g_t)$. Neglecting the effect of $P(g_t)$ is certainly an approximate approach and has to be applied carefully . In our algorithm, we ensure that the approximate approach is consistent during learning and inference. That is, this approximate approach is applied during both the inference and learning processes.

5.5 Experiments

We test our algorithm with a synthetic data set generated from a 2D manifold and embedded in a 3D space as shown in the first row of Figure 5.6. 3000 data points are generated by a 2D random walk, similar to the data set tested in [28], in a rectangle area $[0, 5] \times [-3, 3]$, and then embedded in 3D by a mapping function $f(x, y) = (x, |y|, \sin(\pi y)(y^2 + 1)^{-2} + 0.3y)$. This data set is challenging as it is difficult to estimate the neighborhood structure around the neck where the manifold is folded.

The second and third rows of Figure 5.6 show the results using the method by Roweis's algorithm described in Section 4.3 and our algorithm. We train both models with twelve

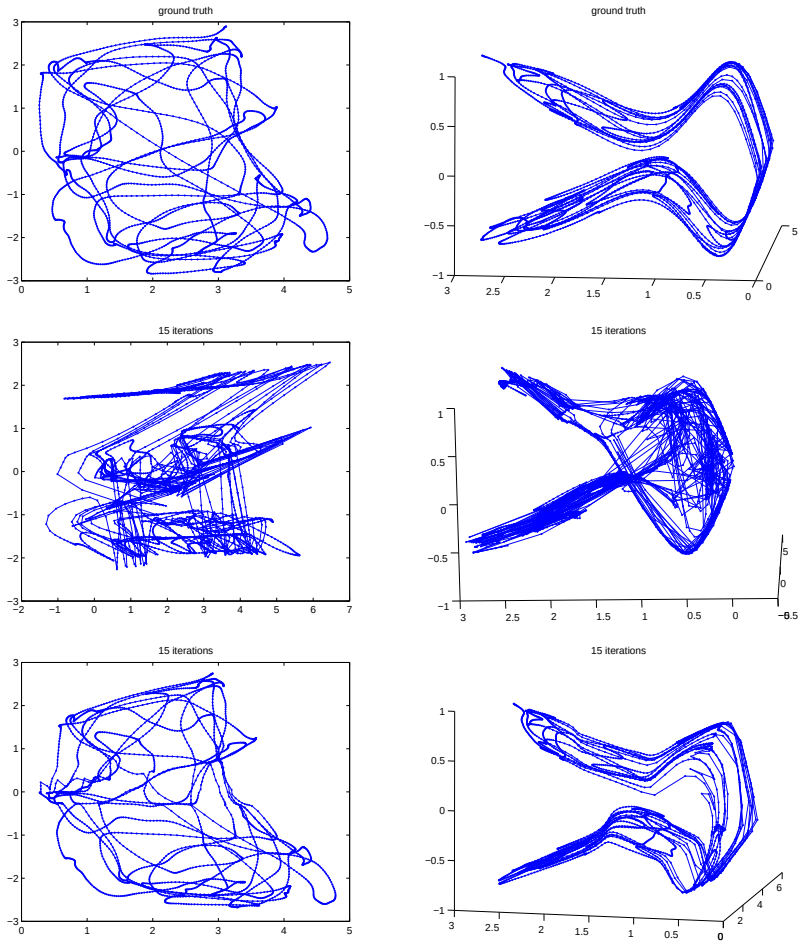

Figure 5.6 The comparison of the global coordination model and our dynamic global coordination algorithm on the synthetic data set.

PPCAs and set the initial conditions to be the same. Note that without taking the temporal information into consideration, the random walk path on the 2D manifold cannot be recovered correctly and thereby the 3D lifted points near the neck region are tangled together. Compared to the ground truth on the first column, our method recovers the 2D manifold better than the unsupervised nonlinear manifold learning algorithm without taking temporal dependence into consideration.

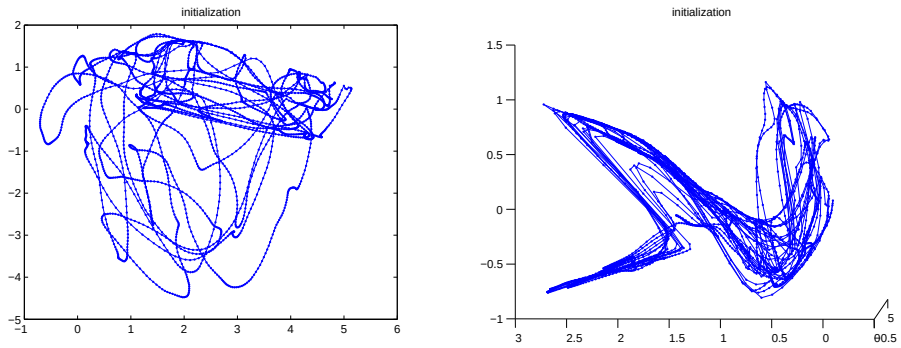

Figure 5.7 The result of the LLC algorithm on the same synthetic data set.

We test the same data set using the LLC algorithm described in Section Sectoin 4.2.2, which learns from i.i.d. samples based on LLE constraints. This result is shown in Figure 5.7. As can be seen, like Roweis's algorithm, it also fails to correctly recover the manifold.

5.6 Summary

In the Chapter, we present a novel manifold learning algorithm that learns from temporal sequences. Our model is based on the global coordination model described in Chapter 4, but we expand it to become a state-space model to take into account temporal dependency among the observed samples. The exact inference is intractable so we present our variational based inference and learning algorithms. The experimental result on the synthetic data set demonstrates that our model apparently achieves better learning result than the global coordination model. In the next Chapter, we will present experiments of applying our algorithm to solve real world learning problems.

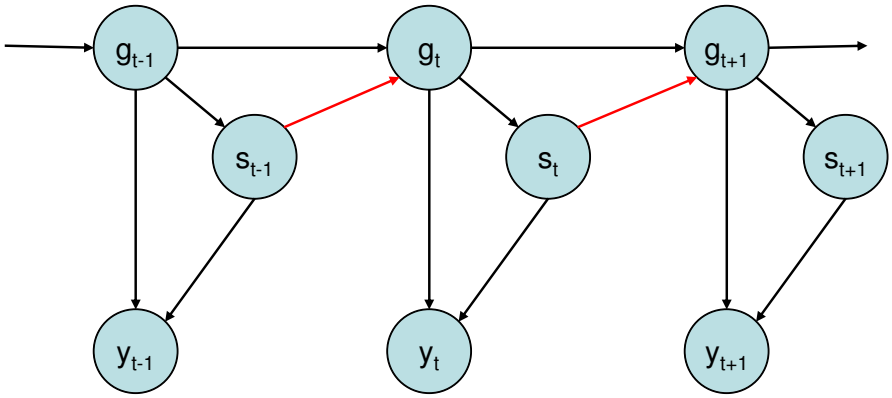

Figure 5.8 Our dynamic global coordination model with nonlinear Markovian dynamics.

Although our algorithm derived in this chapter is based on the assumption of linear dynamics, the model can be extended for a nonlinear dynamic system. By adding an additional link from s_t and g_{t+1} shown in Figure 5.8, our system can model nonlinear dynamics using piecewise linear model. That is, for each given S, $P(y_{t+1}|y_t, s_t = s)$ is a linear dynamic model. This expansion share some resemblance with the switching linear dynamic models such as [59] and [58], and will be capable of modeling a nonlinear dynamic process on a nonlinear manifold.

CHAPTER 6

Applications of Dynamic Global Coordination Model

In the this chapter, we apply our dynamic global coordination model to two real world problems: appearance-based object tracking and robot map learning and localization.

Our first experiment is on appearance-based tracking. The purpose is to demonstrate that our dynamic model can be easily incorporated into a tracking system to track the detail of a dynamic process. We first apply our dynamic global coordination model to learn the appearance model the tracked object from its image sequences. Then, we demonstrate that with our dynamic model we can track the object's location and its appearance variation simultaneously. Since the appearances of the tracked object in the video has strong temporal dependency, tracking the appearance parameters can improve the robustness of the tracker.

In our second experiment which is the focus of this chapter, we apply our model for the landmark-based robot map learning and localization. Here we formulate the map learning and localization problem as a manifold learning problem. We want to acquire a nonlinear mapping that maps the robot's sensory input to a two-dimensional space which represents the robot's location from the sequences of the robot's exploration experiences. In our experiment, the senor is a camera mounted on the robot. Therefore, the orientation

of the camera has to be taken into consideration. This further complicates the learning problem. Our experimental results demonstrate that our dynamic global coordination model successfully solve this mapping learning and localization problem.

6.1 Object Tracking

We apply the proposed dynamic model to the appearance-based tracking problem. It is known that images of an object appearance taken under different illumination conditions and viewing angles are embedded on a low-dimensional nonlinear manifold. In addition, a sequence of such observations is expected to form a smooth trajectory on this manifold. By exploiting this strong temporal dependency, we can reliably track the object in a video. We model this appearance variation of the tracked object using our proposed dynamic global coordination model, and incorporate our model into a standard taker. The graphical model of our tracking system is shown in Figure 6.1 where y_t is the video frame at time t, l_t is the location parameters that specifies the the object's location, and g_t is the appearance parameters of the object at time t.

6.1.1 Rao-Blackwellized Particle Filter

According to Figure 6.1, the state variables now includes the location and the appearance parameters. If both parameters are to be tracked by the particle filter [60], the tracking system will quickly become infeasible as the dimensions of the appearance parameters grow.

Given $P(l_t, g_t|x_{1:t})$, we can factorize the posterior as:

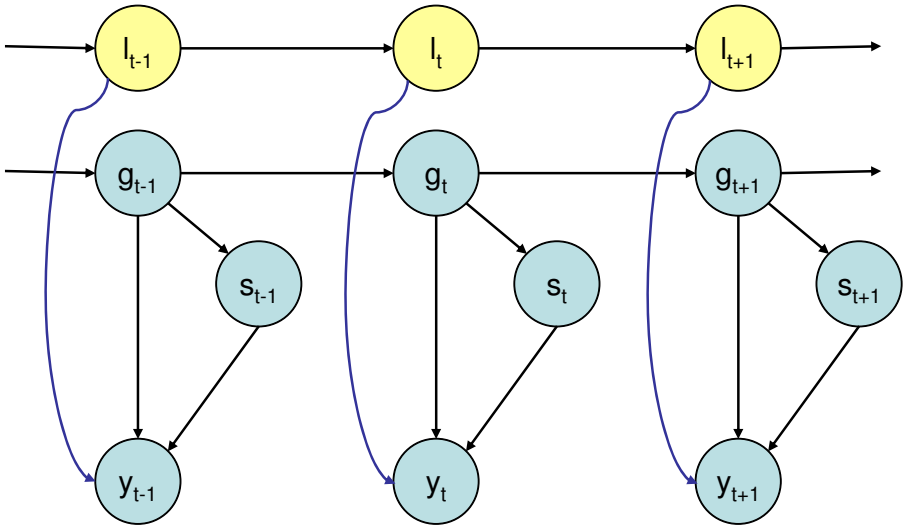

Figure 6.1 Extension of our dynamic global coordination model for object tracking. Based on this model, we apply Rao-Blackwellized particle filter for efficient tracking.

$$P(l_t, g_t|x_{1:t}) = P(g_t|x_{1:t}, l_t)P(l_t|x_{1:t}) \tag{6.1}$$

The inference algorithm in Section 5.3 approximates $P(g_t|x_{1:t}, l_t)$ as an Gaussian distribution. Therefore, our tracker can sample particles only on l_t and model $P(g_t|x_{1:t}, l_t)$ using an analytical distribution. That is, our tracker can use Rao-Blackwellized particle filter (RBPF) [61][62] for efficient tracking.

We test our model on a face tracking experiment which undergoes large pose variations. In our tracking video, there are other faces around the target object. We first test the video using a baseline tracker that tracks location parameters l_t only, and use a mixture of factor analyzers as the measurement function. The result shows that this tracker might track the wrong target when the two faces are close. On the other hand,

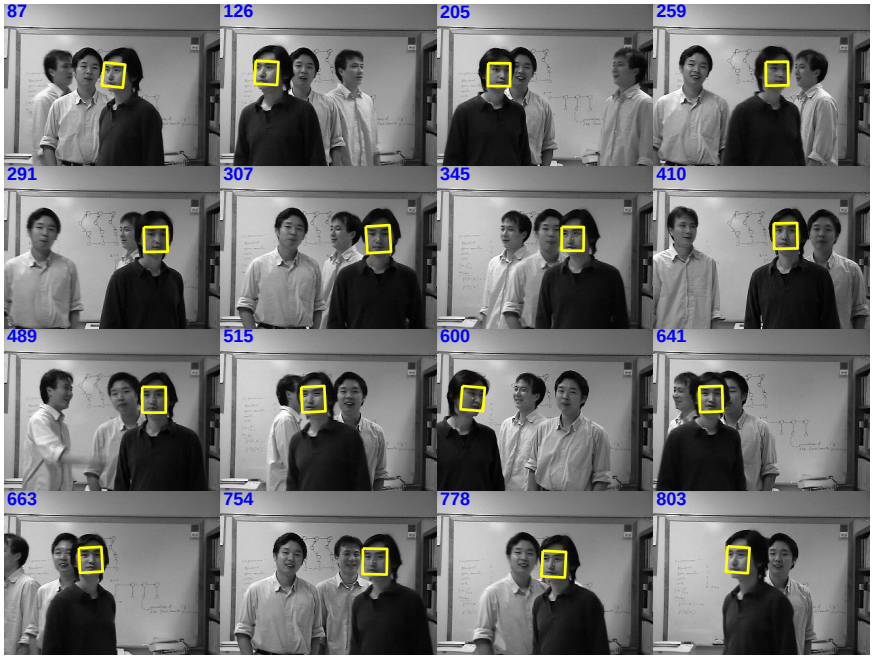

Figure 6.2 Tracking results (left to right on each row): a target with large pose variation and moving in close proximity of similar faces. Our algorithm is able to track the target person in different pose, without confusing with other people.

our tracker is able to track the target well even though several similar objects appear in close proximity because we exploit the temporal dependency in the appearance images of the target (i.e., global coordinates). Figure 6.2 shows the tracking results using the proposed method.

6.2 Robot Map Learning and Localization

Our second experiment is on robot map learning and localization. Acquiring a conceptual map of the environment has been a central problem for mobile robots. Typically, this learning problem is under the framework of simultaneously localization and mapping (SLAM) [63][64], in which the goal is to incrementally reconstruct the 2-D or 3-D space of the environment during the robot's exploration. In SLAM, the sensor used by the robot is

capable of providing geometrical measurement of the relative distance between the robot and its surroundings. Together with the odometer readings which reports the geometrical estimates of the robot's motions, the map is built during the robot's exploration.

Here we adopt a different approach. In our experiment, the sensor input no longer provides the distance measurement. Instead, we have to learn a nonlinear mapping function from the sensory input to the robot's current location. That is, we formulate the robot map learning and localization problem as a manifold learning problem. In addition to the sensory input, the odometer readings on the robot also provide valuable information for the map learning, and we want to include this extraneous information in our model. Learning a manifold from sequences of observations and actions is a relatively new idea, and, to out best knowledge, the only related work is by Bowling et. al. [30] in which they propose a novel algorithm to learn the nonlinear embedding from observation and action sequences.

6.2.1 The Environment

We first introduce our experimental setting and then we formalize our learning problem. Figure 6.3 shows the setting of our mapping learning experiment. We have mobile robot with two video cameras mounted on its head, but here we we use only one of those. The camera collects images of the environment every time the robot performs an action, and the action-image sequence is the data set we used for map building. There are three actions the robot can perform: moving forward, turning left and turning right. Because our odometer measurement is extremely noisy, we can only obtain the information of a

rough estimate of the distance the robot has just traveled and whether the robot has just turned left/right for a unknown angle.

Our map learning and localization algorithm is based on landmarks. As can be seen in Figure 6.3 there are eight landmarks in the environment. To facilitate the learning, these landmarks are chosen such that they can be easily distinguished from each other. However, as will pointed out in the following, our proposed localization method can be extended for situations when landmarks are less distinguishable. For each given image, we detect the landmarks in it, and label them according to their identities. We also record the locations of these landmarks in the image. These are the features used for mapping.

Figure 6.3 The environment of our map learning experiment.

6.2.2 The Learning Problem

Our robot map learning problem is defined as follows. Given an image of the environment taken by the robot, we want to infer the robot's position on the ground where this image is taken. That is, we want to learn a mapping from the extracted features in a image to a three dimensional space (l_x, l_y, h) in which $l = (l_x, l_y)$ is the robot's location and h represents its heading. In our experiment, l is a two-dimensional vector with continuous values. As for the heading h, since both the odometer or the sensory input provide little information to accurate estimate it, h is discretized into 8 values in which two adjacent headings are separated by 45 degree. Figure 6.2.2 illustrates an example of the robot localization result.

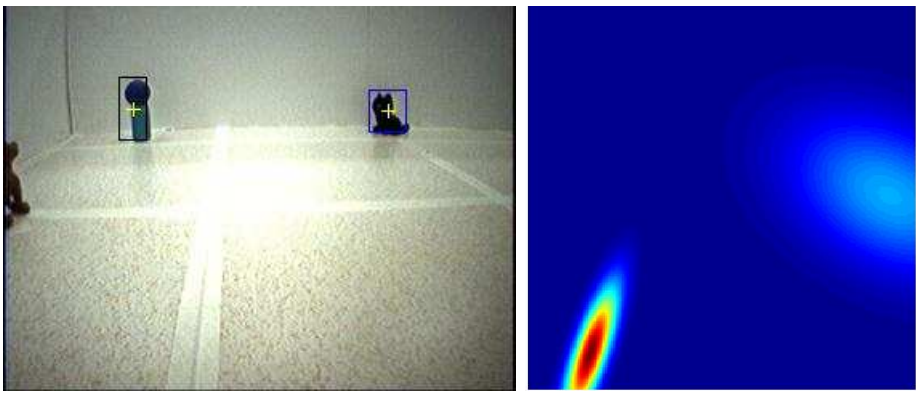

Figure 6.4 An example of the perceived image and the localization result. The figure on the right shows the distribution of l in the 2D given the sensory inputs from the left image. In order to show th result in 2D, we marginalize out h.

With the learning problem well defined, now we can introduce our map model.

6.2.3 The Model

Our model is a hierarchical mixture model, and its structure from top-down is described as follows. At the topmost level, we build a separate mapping function for each landmark. Each mapping functions is actually a mixture of 8 global coordination models, and each global coordination model accounts for one of the 8 possible headings. This is shown in Figure 6.5.

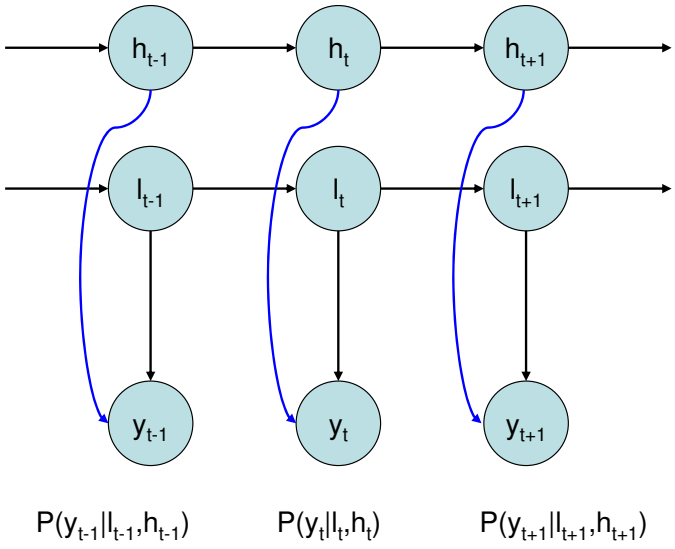

Figure 6.5 Our model for robot map learning and localization. Note that there are 8 global coordination models that provide the mapping between sensory input y_t and location l_t, and h_t determines which global coordination is used.

Let $o \in \{1, \ldots, N\}$ be the index of the landmark, $h \in \{1, \ldots, 8\}$ be the index for the heading and $s \in \{1, 2, 3, 4\}$ be the index of the mixture components in a global coordination model. Denote y^o as the location of the landmark image and (l, h) is the

location and orientation of the robot. Robot localization from a single image is defined as:

$$P(l, h|y^o) = P(h|y^o)P(l|y^o, h) \tag{6.2}$$

For each h we use a global coordination model for $P(l|h, y^o)$,

$$P(l|y^o, h) = \sum_s P(l|y^o, s, h)P(s|y^o, h) \tag{6.3}$$

Note that in the global coordination model the actual distribution of $P(l|y^o, h)$ will be close to a Gaussian, and we approximate it as a single Gaussian using our method described in Section 5.3.1. As a result, $P(l|y^o) = \sum_h P(l, h|y^o)$ in (6.2) is a mixture of 8 weighted Gaussians. Since for a given y^o there are only a few h with nonzero $P(h|y^o)$, the observed mixture is usually less than 8.

We use the global coordination model to describe the nonlinear mapping between y^o and l. That is, we approximate the nonlinear mapping with a mixture of locally linear model. Each local model in the global coordination model provides a linear mapping between y^o and l,

$$y^o = A_s^{(o,h)}(l - \kappa_s^{(o,h)}) + \mu_s^{(o,h)} + v \tag{6.4}$$

where $v \sim \mathcal{N}(0, R)$ is a zero mean Gaussian noise. This is certainly a very crude approximation. Especially when there is an apparent perspective effect. Here we reduce this

inconsistency problem by increase variance in the noise v. With all these defined, we can present our inference and learning algorithms.

6.2.4 Inference

To simplify the analysis, we assume there is one landmark detected in each image frame. Therefore, we neglect variable o in the following derivation.

Given the model, both our inference and learning algorithms are based on the optimal heading sequences $h^*_{1:T}$:

$$h^*_{1:T} = \arg\max_{h_{1:T}} P(h_{1:T}|y_{1:T}, a_{1:T}) \tag{6.5}$$

which can compute by a variant of Viterbi algorithm. Let $\delta_t(i)$ be the likelihood

$$\delta_t(i) = \max_{h_{1:T}} P(h^*_{1:t-1}, h^*_t = i, y_{1:T}, a_{1:T}) \tag{6.6}$$

and denote

$$\phi_i(l_t) = P(l_t|y_{1:t}, a_{1:t}, h^*_{1:t-1}, h_t = i) \tag{6.7}$$

Where $h^*_t = i$, $\phi_i(l_t)$ serves as the prediction of the robot's location given $y_{1:t}$, $a_{1:t}$ and the optimal heading sequence $h^*_{1:t}$.

Starting with $\delta_1(j) = P(h_1 = j|y_1)$ and $\phi_j(l_1) = P(l_1|h_1 = j, y)$. The recursive equation for $\delta_t(j)$ and $\phi_j(l_t)$ can be derived as follows:

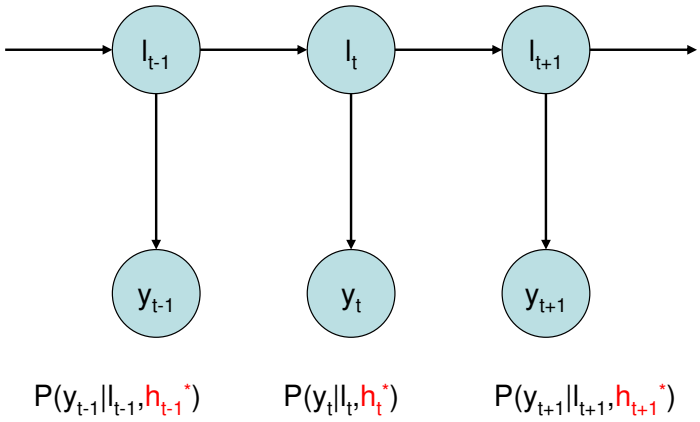

Figure 6.6 The reduced map model when the optimal heading sequence $h^*_{1:T}$ is known. Note that we use the global coordination model indexed by h_i to provide the mapping between g_t and y_t

$$\delta_t(j) =$$

$$\max_i \int_{l_t} P(y_t|h_t = j, l_t) \int_{l_{t-1}} P(l_t|l_{t-1}, a_{t-1}) P(h_t = j|h_{t-1} = i) \phi_i(l_{t-1}) \delta_{t-1}(i) \quad (6.8)$$

$$\phi_j(l_t) = P(y_t|h_t = j, l_t) \int_{l_{t-1}} P(l_t|l_{t-1}, a_{t-1}) \phi_i(l_t), \quad h^*_{t-1} = i. \quad (6.9)$$

With h_t known, (6.8) is computed using our filtering algorithm described in Section 5.3.2. Hence, $\phi_j(l_t) \sim \mathcal{N}(\mu^j_t, \Sigma^j_t)$ is a Gaussian. Note that $\phi_i(l_t)$ is the byproduct while computing $\delta_t(i)$. As a result, when we obtain the optimal sequence $h^*_{i:T}$ we also have $\{\phi_i(l_t), h^*_t = i\}_t$.

6.2.5 Learning

Learning is also based on the optimal heading sequence. With $h^*_{1:T}$, we reduce the map learning model to a dynamic global coordination model as is shown in Figure 6.6. Note that given $h^*_t = i$ the mapping between y_t and g_t is provided by the i-th global coordination model, $P(y_t|g_t, h_t = i)$. It can be proved that $\{\phi_i(l_t), h^*_t = i\}_t$ contains the filtering result of this reduced model, and by using the smoothing algorithm described in Section 5.3.3, we can obtain $\{(\mu^T_t, \Sigma^T_t)\}_t$. The rest procedure are identical to the learning of our dynamic global model. The detailed algorithm is summarized in Table 6.1.

The Variational Map Learning

Given data set $\{y_1, \ldots, y_T\}$, $\{a_1, \ldots, a_T\}$ and the initial model parameters.

The E-step:

1. Compute the optimal sequence $h^*_{1:T}$ and $\{\phi_i(l_t), h^*_t = i\}_t$ using the algorithm descried in Section 6.2.4.
2. Compute $\{(\mu^T_t, \Sigma^T_t)\}_t$ by applying the smoothing algorithm on $\{\phi_i(l_t), h^*_t = i\}_t$.
3. For $h^*_t = i$, update q_{ts} in the i-th global coordination model using (5.35)

The M-step:

1. For $h^*_t = i$, use q_{ts} and $\{(\mu^T_t, \Sigma^T_t)\}_t$ to update the model parameters in the in the i-th global coordination model.

Table 6.1 The variational algorithm for the our map model.

6.2.6 Result

We test our map learning algorithm using the robot's observation-action sequences collected in the environment shown in Figure 6.3, and test the model on new sequences. In our experiment, 10 observation sequences with each sequence have around 100 are

used to train the model. The model is then tested on 3 different testing observation sequences. Figure 6.2.6 and 6.2.6 show the snapshots of the two sequences localization results.

Because the ground truth of the real trajectory is not available, we are unable to perform quantitative analysis on our results. However, by judging from the Figures shown above, the optimal heading sequence $h^*_{1:t}$ is accurate, and the robot's trajectory remain smooth and the predicted locations are reasonable.

6.3 Summary

In this chapter, we apply our model to solve two real world problems: appearance-abased object tracking and robot map learning and localization. In appearance-based object tracking, we demonstrate that our model can be easily incorporated into the tracker to solve more complicated tracking problems. In our case, we track the appearance and location parameters of the object at the same. In addition, since dynamic model always produce an single Gaussian distribution during filtering, we can apply the Rao-Blackwellized particle filter for efficient tracking. We would like to emphasize that our model can be applied to any tracking system with various kinds of features as long as those features are embedding on a manifold.

Our second experiment is on robot map learning, in which we learn the nonlinear mapping function from robot's sensory input to the its position on the ground. This is an difficult problem, and it is especially challenging for us since our odometer on the robot contains large noise. The only reliable information available is the temporal

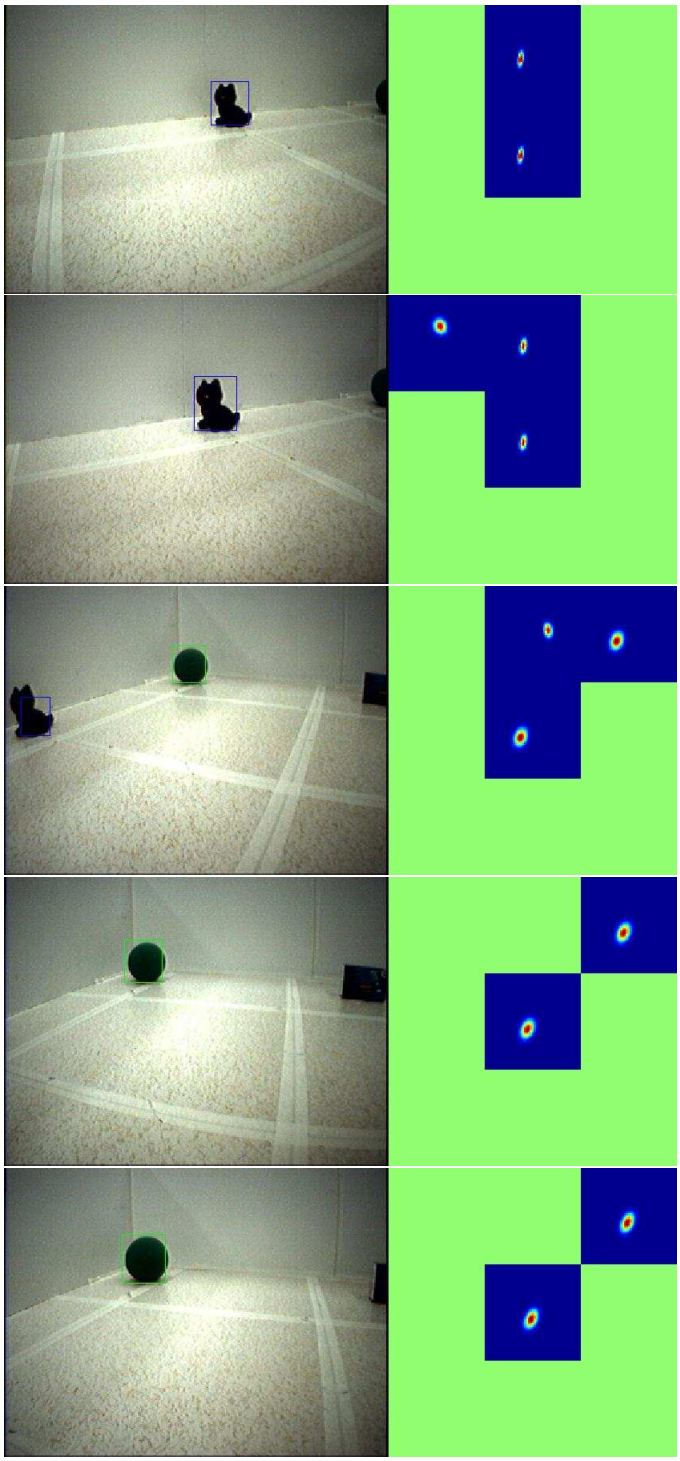

Figure 6.7 An example of the localization result on sequences of images. The box on the right shows 9 distributions. With the center one being $P(l_t|y_1 1:t)$, and the rest 8 represents $P(l_t, h_t|y_1 1:t)$ with adjacent heading aligned close to each other. The green image means $P(h_t|y_1 1:t, h^*_{1:t-1}) = 0$.

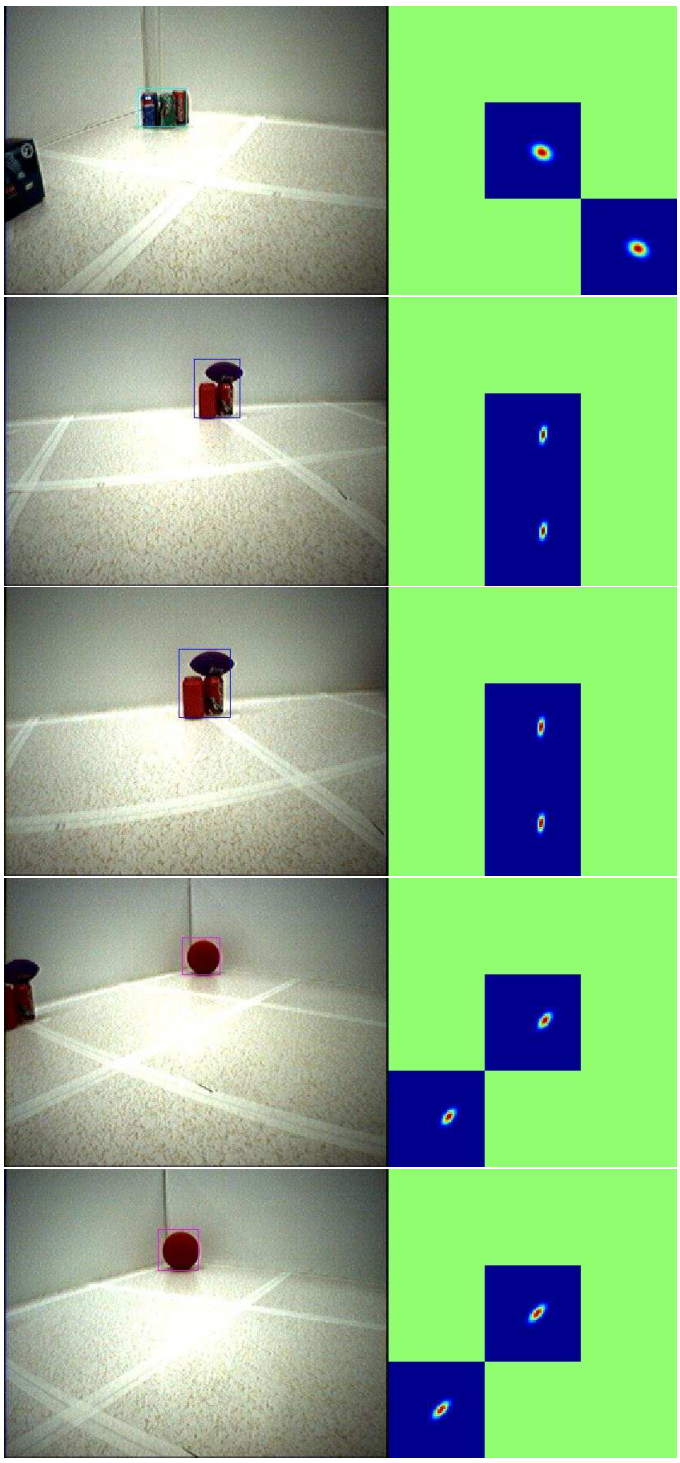

Figure 6.8 Another example of the localization result on sequences of images.

dependency in the observation sequences. Yet, our result demonstrates that we are capable of recovering this mapping. Currently our model is trained with limited size of training sequences. We plan to train our model with more examples and provide more detail quantitative analysis in the near future.

CHAPTER 7

Conclusion

In this thesis, we present a novel algorithm for nonlinear manifold learning from time series, and we demonstrate that the temporal information can greatly improve manifold learning. Our algorithm is based on the global coordination model which uses piecewise linear models to fit a nonlinear manifold, and aligns the parameters in each linear model in a global coordination system. Alignment of these linear models is difficult when i.i.d. samples are sued. We show that a superior learning result is achieved if the global coordination model is trained from temporal sequences. This is done by extending the global coordination model along the time domain to become a dynamic Bayesian network. We then provide an variational approximate inference and learning algorithm for it. Experiments of our algorithm on synthetic and real world data demonstrate the practicability of our model in solving real world problems.

Based on our work in the thesis, there are two major directions we would like to explore:learning our dynamic global coordination online and modeling nonlinear dynamics. Online learning is critical for applications such as the robot map learning. In Section we have presented our online learning algorithm to adaptively adjust a single linear model. We would like to explore the possibility of extending this algorithm to learn the global

coordination model. The second extension is to expand our model to modeling the sequences with nonlinear dynamics. In Chapter 5, we derive our learning algorithm based on the linearly dynamic assumption. However, as we have laid out in Figure 5.8 that our model be extended to model nonlinear dynamic. We are currently deriving the learning algorithm for this model.

Our model presented in this thesis has advantages over existing manifold learning algorithms. We believe that we would find more applications for it in the future.

REFERENCES

[1] C. Bregler and S. M. Omohundro, "Nonlinear manifold learning for visual speech recognition," in *Proceedings of the First IEEE International Conference on Computer Vision*, 1995, pp. 494–499.

[2] R. D.Dony and S. Haykin, "Optimally adaptive transfer coding," *IEEE trans. on Image Processing*, vol. 4, no. 10, pp. 1358–1370, 1995.

[3] Geoffery E. Hinton, Peter Dayan, and Michael Revow, "Modeling the manifolds of images of handwritten digits," *IEEE trans. on Neural Networks*, vol. 8, no. 1, pp. 65–74, January 1997.

[4] C. Tomasi, S. Petrov, and A. Sastry, "3d tracking = classification + interpolation," in *Proceedings of the First IEEE International Conference on Computer Vision*, 2003, pp. 1441–1448.

[5] Michael E. Tipping and Christopher M. Bishop, "Mixtures of probabilistic principle component analyzers," *Neural Computation*, vol. 11, pp. 443–482, 1999.

[6] Qiang Wang, Guangyou Xu, and Haizhou Ai, "Learning object intrinsic structure for robust visual tracking," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003, vol. 2, pp. 227–234.

[7] I.T. Jolliffe, *Principal Component Analysis*, Springer, New York, 1986.

[8] Z. Ghahramani and G. Hinton, "The em algorithm for mixtures of factor analyzers.," Tech. Rep. CRG-TR-96-1, University of Toronto Technical Report, 1996.

[9] T. Hastie and W. Stuetzle, "Principal curves," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502–516, June 1989.

[10] M. LeBlanc and R. Tibshirani, "Adaptive principal surfaces," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 53–64, March 1994.

[11] Neil D. Lawrence, "Gaussian process latent variable models for visualization of high dimensional data," in *Advances in Neural Information Processing Systems*, 2005, number 16.

[12] B. Scholkopf, A. Smola, and K.R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319, 1998.

[13] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, December 2000.

[14] Sam T. Roweis and Lawrence L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, December 2000.

[15] M. Belkin and P. Niyogi, "Laplacian eigenmap and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, 2002, number 14, pp. 585–591.

[16] David L. Donoho and Carrie Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," in *Proceedings of National Academic of Sciences of USA*, 2003, vol. 100, pp. 5591–5596.

[17] Y. Bengio, J. Paiement, and P. Vincent, "Out-of-sample extensions for lle, isomap, mds, eigenmap, and spectral clustering," in *Advances in Neural Information Processing Systems*, 2004, number 16, pp. 585–591.

[18] A. Elgammal and C.S. Lee, "Inferring 3d body pose from silhouettes using activity manifold learning," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[19] Sam Roweis, Lawrence K. Saul, and Geoffery E. Hinton, "Global coordination of local linear models," in *Advances in Neural Information Processing Systems*, 2002, number 14, pp. 889–896.

[20] Matthew Brand, "Charting a manifold," in *Advances in Neural Information Processing Systems*, 2003, number 15.

[21] Y.W. Teh and Sam Roweis, "Automatic alignment of local representations," in *Advances in Neural Information Processing Systems*, 2003, number 15.

[22] Jakob J. Verbeek, Sam Rowies, and Nikos Vlassis, "Non-linear cca and pca by alignment of local models," in *Advances in Neural Information Processing Systems*, 2003, number 15.

[23] C. M. Bishop, G. E. Hinton, and I. D. Strachan, "Gtm through time," in *IEE Fifth International Conference on Artificial Neural Networks*, 1997, number 111-116.

[24] C. M. Bishop, M. Svensen, and C. Williams, "Gtm: the generative topographic mapping," *Neural Computation*, vol. 10, pp. 215–234, 1998.

[25] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biologocal Cybernetics*, vol. 43, pp. 59–69, 1982.

[26] M. Brand, "Shadow puppetry," in *Proceedings of the First IEEE International Conference on Computer Vision*, 1999, pp. 1237–1244.

[27] L.R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[28] A. Rahimi, B. Recht, and T. Darrell, "Learning appearance manifolds from video," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 868–875.

[29] O. C. Jenkins and M. J. Mataric, "A spatio-temporal extension to isomap nonlinear dimension reduction," in *Proceedings of International Conference on Machine Learning*, 2004, pp. 441–448.

[30] M. Bowling, A. Ghodsi, and D. Wilkinson, "Action respecting embedding," in *Proceedings of International Conference on Machine Learning*, 2005.

[31] J.M. Wang, D.J. Fleet, and A. Hertzmann, "Gaussian process dynamic models," in *Advances in Neural Information Processing Systems*, 2005.

[32] Gene H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins University Press, Baltimore, 1996.

[33] C.M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.

[34] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Algorithms*, MIT Press, Cambridge, MA, 1996.

[35] L.K. Saul and S.T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifolds," *Journal of Machine Learning Research*, vol. 4, pp. 119–155, 2004.

[36] H. Zha and Z. Zhang, "Isometric embedding and continuum isomap," in *Proceedings of International Conference on Machine Learning*, 2003.

[37] M.H.C. Law, N. Zhang, and A.K. Jain, "Nonlinear manifold learning for data stream," in *Proceedings of SIAM data mining*, 2004.

[38] K. Weinberger and L. Saul, "Unsupervised learning of image manifolds by semidefinite programing," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2004, pp. 988–995.

[39] S. Schaal and C.G. Atkeson, "Constructive incremental learning from local information," *Neural Computation*, vol. 10, pp. 2047–2084, 1998.

[40] Michael Jordan, Zoubin Ghahramani, Tommi Jaakkola, and Lawrence Saul, "An introduction to variational methods for graphical models," *Machine Learning*, vol. 37, pp. 183–233, 1999.

[41] Michael E Tipping and Christopher M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society, series B*, vol. 61, no. 3, pp. 611–622, 1999.

[42] M. W. Berry, S. T. Dumais, and G. W. O'brien, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, no. 4, pp. 573–595, 1995.

[43] A. Levy and M. Lindenbaum, "Sequential karhunen-loeve basis extraction and its application to images," *IEEE trans. on Image Processing*, vol. 9, no. 8, pp. 1371–1374, 2000.

[44] M. Brand, "Incremental singular value decomposition of uncertain data with missing values," in *Proceedings of European Conference on Artificial Intelligence*, 2002, vol. 1.

[45] S. Roweis, "Em algorithm for pca and spca," in *Advances in Neural Information Processing Systems*, 1997, number 9, pp. 626–632.

[46] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, Wiley-Interscience, New York, 2 ed., 2001.

[47] A.P.Dempster, N.M. Laird, and D.B. Rubin, "Maximum-likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistics Society Series B*, vol. 39, 1977.

[48] D. Ross, J. Lim, and M.-S. Yang, "Adaptive probabilistic visual tracking with incremental subspace update," in *Proceedings of European Conference on Artificial Intelligence*, 2004, vol. 2.

[49] K.-C. Lee J. Ho, M.-H. Yang, and D. Kriegman, "Visual tracking using learned subspaces," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2004.

[50] J. Lim, D. Ross, R.-S. Lin, and M.-S Yang, "Incremental learning for visual tracking," in *Advances in Neural Information Processing Systems*, 2005, number 17.

[51] R.-S. Lin, D. Ross, J. Lim, and M.-S Yang, "Adaptive discriminative generative model and its applications," in *Advances in Neural Information Processing Systems*, 2005, number 17.

[52] P. Hall, D. Marshall, and R. Martin, "Incremental eigenanalysis for classification," in *In Proceedings of British Machine Vision Conference*, 1998, number 286-295.

[53] R.-S. Lin, M.-S Yang, and S.E. Levinson, "Object tracking using incremental fisher discriminant analysis," in *International Conference on Pattern Recognition*, 2004, number 17, pp. 757–760.

[54] R.-S. Lin, M.-S Yang, and S.E. Levinson, "Adaptive discriminative generative model for object tracking," 2004, number 16, pp. 505–509.

[55] J. Kim, "Dynamic linear models with markov switching," *Journal of Econometrics*, vol. 60, pp. 1–22, 1994.

[56] S.L. Lauritzen, *Graphical Models*, Oxford University Press, New York, 1996.

[57] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic Press, 1988.

[58] Zoubin Ghahramani and Geoffrey Hinton, "Variational learning for switching state-space models," *Neural Computation*, vol. 12, pp. 831–864, 2000.

[59] V. Pavlovic, J. Rehg, T.-J Cham, and K. Murphy, "A dynamic bayesian network approach to figure tracking using learned dynamic models," in *CVPR*, 1999, pp. 94–101.

[60] Michael Isard and Andrew Blake, "Condensation-conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[61] Arnaud Doucet, Nando de Freitas, Kevin Murphy, and Stuart Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of Uncertainty in AI*, 2000.

[62] Z. Khan, T. Balch, and F. Dellaert, "A rao-blackwellized particle filter for eigen tracking," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 204.

[63] Randall Smith, Matthew Self, and Peter Cheeseman, "Estimating uncertain spatial relationships in robotics," *Autonomous Robot Vehicles*, pp. 167 – 193, 1990.

[64] C.-C Wang, C. Thrope, and S. Thrun, "Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a gound vehical in crowded urbana areas," in *International Conference on Robotics and Automation*, 2003.

AUTHOR'S BIOGRAPHY

Ruei-Sung Lin received his BS degree in computer science and information engineering from the National Taiwan University in 1994. He attended the Department of Computer Science at the University of Illinois at Urbana-Champaign in 1997, and was a member of Prof. Stephen Levinson's research group from 1998 to 2005. He is currently a senior software engineer at Motorola Labs. His research interests include computer vision, statistical modeling, machine and robot learning.